Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Moderní JavaScript frameworky pro vývoj Single Page Aplikace

Bc. Jana Turoňová

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Jana Turoňová

Informatika

Název práce

Moderní JavaScript frameworky pro vývoj Single Page Aplikace

Název anglicky

Modern JavaScript frameworks for Single Page Application development

---

Cíle práce

Hlavním cílem práce je, na základě provedené komparace, vytvoření doporučení pro využití zvolených frameworků při vývoji webové aplikace využívající architekturu SPA.

Dílčími cíli práce jsou návrh UI specifikace webové aplikace a následná implementace designu na základě vytvořené specifikace.

Metodika

Diplomová práce je založena na studiu odborných informačních zdrojů a použití moderních vývojových nástrojů pro tvorbu webových aplikací.

V teoretické části práce budou popsány využité technologie, a analyzovány a porovnávány frontend JavaScript frameworky.

V praktické části na základě získaných znalostí JavaScript frameworků bude vytvořena implementace osobní stránky. Jednotlivé webové aplikace budou následně testovány a porovnány dle standartních webových metrik.

Doporučený rozsah práce

60-80 stránek

Klíčová slova

Javascript, framework, Angular, React, Vue

Doporučené zdroje informací

AMBLER, Tim a Nicholas CLOUD. JavaScript frameworks for modern web dev. New York: Apress, [2015]. Expert's voice in Web development. ISBN 14-842-0663-0.

CASTRO, E. – HYSLOP, B. *HTML5 a CSS3 : názorný průvodce tvorbou WWW stránek.* Brno: Computer Press, 2012. ISBN 978-80-251-3733-8.

GREIF, Sascha a Raphaël BENITTE. The State of JavaScript 2019 [online]. © 2019, 23.01.2019 [cit. 2022-01-01]. Dostupné z: https://2019.stateofjs.com/

HAVERBEKE, Marijn. Eloquent JavaScript: a modern introduction to programming. 3rd ed. No Starch Press, 2018. ISBN 9781593279516.

PEHLIVANIAN, Ara a Don NGUYEN. JavaScript okamžitě. 2. vydání. Brno: Computer Press, 2021. Expert's voice in Web development. ISBN 978-80-251-5025-2.

Předběžný termín obhajoby

2021/22 LS – PEF

Vedoucí práce

Ing. Petr Hanzlík, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 26. 11. 2022

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 28. 11. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 30. 03. 2023

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Moderní JavaScript frameworky pro vývoj Single Page Aplikace" jsem vypracovala samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autorka uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušila autorská práva třetích osob.

V Praze dne 30.3.2023 _____

Poděkování

Ráda bych touto cestou poděkovala Ing. Petru Hanzlíkovi, Ph.D. za ochotu, podporu a vedení mé diplomové práce.

Moderní JavaScript frameworky pro vývoj Single Page Aplikace

Abstrakt

Tato diplomová práce se zaměřuje na moderní JavaScriptové frameworky vhodné pro vývoj webových single page aplikací (SPA). Teoretická část práce popisuje možné přístupy k tvorbě webových aplikací se zaměřením na technologie a architektury. Speciální pozornost je pak kladena na tři JavaScriptové frameworky pro vývoj webových aplikací - Vue, React a Angular a jejich vzájemné porovnání. V empirické části práce je vytvořeno uživatelské a grafické rozhraní webové stránky a je navržena struktura a architektura SPA projektu. Tato stránka je následně implementována za využití frameworků Vue, React a Angular. Pro každou implementaci je provedeno měření výkonových a UX metrik, jak pro mobilní tak i pro desktopová zařízení, dále bylo provedeno měření spotřeby zdrojů. Frameworky jsou analyzovány také z hlediska strmosti křivky učení, podpory prohlížečů a velikosti komunity. Ve výsledcích jsou uvedena doporučení pro použití těchto frameworků na základě výsledků komparace, dle potřeby aplikace a dle zkušeností vývojáře.

Klíčová slova: JavaScript, framework, Angular, React, Vue, Single Page Application

Modern JavaScript frameworks for Single Page Application development

Abstract

This thesis focuses on modern JavaScript frameworks suitable for development of single page web applications (SPA). The theoretical part of the thesis describes the possible approaches to creating web applications, with a focus on technologies and architectures. Special attention is then paid to three JavaScript frameworks for developing web applications - Vue, React and Angular and their mutual comparison. In the empirical part of the work, the user and graphic interface of the website is created, and the structure and architecture of the SPA project is designed. This proposed page is subsequently implemented using the Vue, React and Angular frameworks. Performance and UX metrics are assessed for each implementation, both for mobile and desktop devices, resource consumption of individual implementation was measured too. Frameworks are also analysed in terms of learning curve steepness, browser support and community size. The conclusions include recommendations for the use of these frameworks based on the results of the comparison, according to the needs of the application and according to the developer's experience.

Keywords: JavaScript, framework, Angular, React, Vue, Single Page Application

Obsah

1 Úvod 12

2 Cíl práce a metodika 13

2.1 Cíl práce 13

2.2 Metodika 13

3 Teoretická východiska 14

3.1 Programovací paradigma 14

3.1.1 Imperativní programování 14

3.1.2 Funkcionální programování 15

3.2 Základní kameny webu 15

3.2.1 HTML5 15

3.2.1 Markdown 16

3.2.2 CSS3 17

3.3 Design 20

3.3.1 Statický design 20

3.3.2 Dynamický design 20

3.4 SEO 22

3.5 Webové aplikace 23

3.5.1 Multi-page applications (MPA) 23

3.5.2 Single-page applications (SPA) 24

3.6 Datové architektury 25

3.6.1 MVC 25

3.6.2 MVP 26

3.6.3 MVVM 27

3.7 DOM 27

3.7.1 Virtuální DOM 28

3.7.2 Přírůstkový DOM 29

3.8 JavaScript 29

3.8.1 Historie JavaScript 30

3.8.2 Vanilla JavaScript 31

3.9 TypeScript 32

3.10 JavaScript Framework 33

3.10.1 React 33

3.10.2 Vue 35

3.10.3 Angular 38

3.11 Teoretické porovnání JS Frameworků 39

3.11.1 Výsledky ankety The State of JS 40

3.11.2 Metriky front-end frameworků 41

3.12 Node.js ... 43
3.12.1 Architektura Node.js ... 43
3.12.2 Výhody a Nevýhody Node.js ... 46
3.13 Zprovoznění aplikace ... 46
3.13.1 Doména ... 46
3.13.2 Hostitel ... 47
3.13.3 Přesun aplikace na server ... 48
4 Vlastní práce ... 49
4.1 Uživatelské a grafické rozhraní ... 50
4.1.1 Informační architektura ... 50
4.1.2 Wireframe ... 50
4.1.3 Hlavička (header) ... 55
4.1.4 Tělo (body) ... 55
4.1.5 Patička (footer) ... 55
4.1.6 Grafický vzhled ... 55
4.2 Struktura projektu ... 56
4.2.1 Client-side ... 56
4.2.2 Server-side ... 56
4.2.3 Propojení Server-side a Client-side ... 56
4.2.4 API server ... 57
4.2.5 Spuštění aplikace ... 57
4.3 Testování JS frameworků ... 57
4.3.1 Použité metriky ... 58
4.3.2 React ... 60
4.3.3 Angular ... 63
4.3.4 Vue ... 65
4.4 Porovnání rozdílu mezi jednotlivými frameworky ... 68
4.4.1 Mobilní verze ... 68
4.4.2 Desktopová verze ... 68
4.5 Analýza porovnání vybraných frameworků ... 70
4.5.1 Podpora prohlížečů a velikost komunity ... 70
4.5.2 Křivka učení ... 70
4.5.3 Měření spotřeby zdrojů ... 71
5 Výsledky ... 72
5.1 Potřeby aplikace ... 72
5.1.1 Doporučení pro vytvořenou aplikaci ... 72
5.1.2 Doporučení dle velikosti aplikace ... 72
5.2 Zkušenosti vývojáře ... 73
6 Závěr ... 75

7 Seznam použitých zdrojů 76

8 Přílohy 80

8.1 Ukázka webové stránky 80

8.2 Měření 83

8.2.1 React – mobilní verze 83

8.2.2 Vue – mobilní verze 84

8.2.3 Angular – mobilní verze 85

8.2.4 React – desktopová verze 86

8.2.5 Vue – desktopová verze 87

8.2.6 Angular – desktopová verze 88

Seznam obrázků

Obrázek 1 - Schéma MVC (24) (25) 25

Obrázek 2 - Schéma MVP (23) (25) 26

Obrázek 3 - Schéma MVVM (25) (26) 27

Obrázek 4 - HTML DOM – Document Object Model (27) 28

Obrázek 5 - React Router (vlastní zpracování) 35

Obrázek 6 - Vue Router (vlastní zpracování) 37

Obrázek 7 - Benchmark test Vue, React, Angular 1/2 (45) 41

Obrázek 8 - Benchmark test Vue, React, Angular 2/2 (45) 42

Obrázek 9 - Doba spuštění (45) 42

Obrázek 10 - Doba alokace paměti (45) 43

Obrázek 11 - Ukázka kódu v Node.js, (vlastní zpracování) 45

Obrázek 12 - Informační architektura (vlastní zpracování) 50

Obrázek 13 - Návrh webové stránky pro mobilní zařízení (vlastní zpracování) 51

Obrázek 14 - První wireframe – desktopová verze (vlastní zpracování) 52

Obrázek 15 - Druhý wireframe – desktopová verze (vlastní zpracování) 52

Obrázek 16 - Návrh webové stránky pro mobilní zařízení (vlastní zpracování) 53

Obrázek 17 – Třetí wireframe – desktopová verze (vlastní zpracování) 54

Obrázek 18 - Čtvrtý wireframe – desktopová verze (vlastní zpracování) 54

Obrázek 19 - Podrobnosti běhu testu PageSpeed (vlastní zpracování) 58

Obrázek 20 - Výstup z PageSpeed (vlastní zpracování) 59

Obrázek 21- Domovská stránka (vlastní zpracování) 80

Obrázek 22 – Galerie (vlastní zpracování) 80

Obrázek 23 - Galerie – zvětšení obrázku (vlastní zpracování) 81

Obrázek 24 - Hra TicTacToe (vlastní zpracování).....81
Obrázek 25 – Resume (vlastní zpracování).....82

Seznam tabulek

Tabulka 1 - Verze ECMA-262 (7) (32).....31
Tabulka 2 - Metriky pro React pro mobilní verzi(vlastní zpracování) 61
Tabulka 3 - Metriky pro React pro desktopovou verzi(vlastní zpracování) 62
Tabulka 4 - Metriky pro Angular pro mobilní verzi (vlastní zpracování) 64
Tabulka 5 - Metriky pro Angular pro desktopovou verzi (vlastní zpracování) 65
Tabulka 6 - Metriky pro Vue pro mobilní verzi(vlastní zpracování).....66
Tabulka 7 - Metriky pro Vue pro desktopovou verzi(vlastní zpracování).....67
Tabulka 8 - Porovnání komunity, podpory prohlížečů a spotřeby zdrojů (vlastní zpracování).....71

Seznam použitých zkratek

WWW	World Wide Web
SPA	Single-page Application
UI	Uživatelské rozhraní
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
SASS	Syntactically Awesome Style Sheets
SCSS	Sassy Cascaded Style Sheets
tzn.	to znamená
tzv.	takzvaný
JS	JavaScript

1 Úvod

První veřejně dostupná webová stránka, jejímž obsahem byl popis projektu WWW, vznikla 6. srpna 1991, což je víc jak před 30 lety, a jako autor této webové stránky se uvádí Tim Berners-Lee, který ji zveřejnil v diskusní skupině alt.hypertext. Bohužel, podoba této webové stránky se nezachovala. (1)

V průběhu dalších 30 let se technologie velmi rychle rozvinuly a vzrostl i počet uživatelů internetu. Uživatelé v dnešní době přistupují k webovým aplikacím nejen z desktopového počítače, ale i chytrých telefonů, laptopů, tabletů a dalších zařízení. Nynější webové aplikace kladou důraz na uživatelskou přívětivost a interakci s uživatelem.

Vývoj webových aplikací se v průběhu let začal rozlišovat na frontend a backend. Důvodem je využití různých technologií, kdy technologie pracují buď na straně serveru nebo na straně klienta, označovaného jako prohlížeč. Webové aplikace dokážou fungovat bez použití backend tedy pouze s frontend, avšak bez znalosti frontend se při vývoji webové aplikace vývojáři neobejdou.

V dnešní době se nevyplatí časově ani nákladově psát čisté kódy, protože vznikly frameworky, které usnadňují práci, a tak není potřeba se zabývat základními konstrukcemi jazyka. Nejznámější front-end frameworky jsou CSS frameworky, a právě JavaScriptové frameworky. Mezi nejznámější moderní JavaScript frameworky spadají Angular, React a Vue.

Všechny tyto front-end JavaScript frameworky jsou ale různé svou architekturou a strukturou, proto je nutné nejprve pochopit jejich princip fungování a rozdílnost mezi danými JavaScript frameworky.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem práce je, na základě provedené komparace, vytvoření doporučení pro využití zvolených frameworků při vývoji webové aplikace, využívající architekturu SPA.

Dílčími cíli práce jsou návrh UI specifikace webové aplikace a následná implementace designu na základě vytvořené specifikace.

2.2 Metodika

Diplomová práce bude založena na studiu odborných informačních zdrojů a použití moderních vývojových nástrojů pro tvorbu webových aplikací.

V teoretické části práce budou popsány využité technologie, a analyzovány a porovnávány front-end JavaScript frameworky.

V praktické části na základě získaných znalostí JavaScript frameworků bude vytvořena implementace osobní stránky. Jednotlivé webové aplikace budou následně testovány a porovnány dle výkonnostních webových metrik a bude provedeno měření spotřeby zdrojů. V potaz budou brány i další kritéria jako je například podpora prohlížečů a velikost komunity. Na základě těchto kritérií bude sestaveno doporučení pro využití jednotlivých zvolených frameworků při vývoji webové aplikace.

3 Teoretická východiska

3.1 Programovací paradigma

Programovací paradigma popisuje, jaký styl byl použit k psaní a organizaci programového kódu. Existuje velké množství paradigmat, zde jsou ty nejdůležitější v rámci JavaScriptu: (2)

- Imperativní programování,
    - Procedurální programování,
    - Objektově orientované programování,
- Funkcionální (deklarativní) programování.

3.1.1 Imperativní programování

Imperativní programování se zaměřuje na pohled – „Jak to udělat.“ Jedná se o sadu instrukcí, která bude říkat počítači, jak má věci dělat, kód je založen na definování proměnných a práci s nimi. Mezi imperativní programování patří známé procedurální a objektově orientované programování, též OOP. (2) (3)

Procedurální programování

Nejstarší způsob zápisu programovacího kódu. Tento způsob zápisu kódu připomíná návod k montáži nábytku. Obsahuje jednotlivé kroky, které se musí provést abychom dosáhli požadovaného výsledku. (2) (4)

Objektově orientované programování

Jak již název napovídá jedná se o programování pomocí objektů, dále tříd a prototypů. Hlavní myšlenkou tohoto přístupu je mít objekty jako programátorem poskytovanou datovou strukturu, která může být manipulována metody.

U implementace třídy vycházející z jiné třídy se propojují vlastnosti a chování, tomu se říká dědičnost. Hlavní výhodou tohoto typu přístupu je, že se eliminuje redundance kódu. (2) (4)

3.1.2 Funkcionální programování

Jedná se o poddruh tzv. deklarativní programování. Paradigma založené na volání funkcí, které jsou tzv. vyššího řádu, tzn. že se můžou používat i jako parametry nebo naopak funkce může vracet další funkci. Důležitým pravidlem je že vrací stejné výstupy na stejné vstupy.

U JavaScriptu se nejedná o plně funkcionální programování, ačkoliv se poslední dobou velmi prosazuje.

Příklady funkcionálních programovacích jazyků jsou LISP (1958), Haskell (1990) a F# (2005). (2) (3)

3.2 Základní kameny webu

Základy webových aplikací tvoří nejznámější značkovací jazyk HTML a kaskádový styl CSS, od kterých se následně odvíjí Frameworky a programovací jazyky.

3.2.1 HTML5

Hypertext Markup Language, známý pod zkratkou HTML, je značkovací jazyk používaný pro tvorbu webových stránek zpracováván klientem. Při načtení stránky vytváří klient strom objektů, který představuje celý dokument (DOM). Tento značkovací jazyk popisuje strukturu webových stránek pomocí značek například záhlaví, odstavce a seznamy. (5)

> *„Jazyk HTML vznikl na počátku 90. let minulého století v podobě stručného dokumentu popisujícího několik elementů používaných pro tvorbu webových stránek.“ (6)*

HTML se stále vyvíjí, momentálně nejnovější verze jazyka HTML je HTML5. Číslo za HTML určuje verzi jazyka, jak se jazyk přizpůsoboval a rozrůstal o nové elementy. Například u HTML5 byly přidány nové elementy header, footer, main, nav, article, section a další. Prvky Header, main a footer jsou stanoveny jako základní elementy pro hlavní element body, pro ukázku jejich umístění je zde sestaven kód a podrobný popis je poskytnut níže. (6) (5)

```
<!DOCTYPE html>
<html lang="en">

        <head>
           <meta charset="UTF-8">
           <meta http-equiv="X-UA-Compatible" content="IE=edge">
           <meta name="viewport" content="width=device-width, initial-scale=1.0">
        </head>

        <body>
           <header>
           </header>
           <main>
           </main>
           <footer>
           </footer>
        </body>
        <script scr=“first.js“> </script>
</html>
```

Každý HTML dokument je vybaven deklarací DOCTYPE, z toho důvodu, aby prohlížeč věděl, jaký typ dokumentu má očekávat. V konkrétním příkladu je použita deklarace pro HTML5. Následně je specifikován jazyk, ve kterém bude stránka napsána.

Head, hlavička dokumentu, obsahuje metadata, tj. informace o datech. V této části se nachází například název stránky, použitá znaková sady, styly a další meta informace.

Body, tělo, nejdůležitější část HTML dokumentu. Obsahuje veškerý obsah HTML dokumentu jako jsou nadpisy, odstavce, tabulky, články, hypertextové odkazy a další. (6) (7)

Následuje element script, důvodem vkládání scriptu těsně nad element </body> je předejít chybám, které by mohly vzniknout. Pokud by byl element script v elementu head, kód v scriptu by se spustil okamžitě a odkazoval na elementy v body, ještě před načtením všech elementů v body, což by vedlo k chybě. (8)

3.2.2 Markdown

Markdown je zjednodušený značkovací jazyk, který vyvinul John Gruber v roce 2004 a je jedním z populárních značkovacích jazyků. (9)

Pro psaní markdownu může být použit program Visual Studio Code s instalovaným pluginem. Mohou být využity i jiné programy, které byly speciálně navrženy pro psaní

Markdownu jako je například Typora, Dillinger a další. Přípona souboru je vždy .md nebo .markdown. (9)

Pro přidání Markdownu do webové stránky je potřeba použít nástroj, který převede Markdown syntaxi na HTML kód. V závislosti na použití jazyka pro webové stránky je nutné dohledat vhodné knihovny a nástroje, které poskytují převod. Nástroje a knihovny mohou být součástí nejrůznějších frameworků nebo mohou být instalovány jako samostatné balíčky jako například marked. Zde je ukázka HTML kódu a Markdownu. (10)

```
<h1> Nadpis 1</h1>
<h2> Nadpis 2</h2>

Takto vypadá <b> tučný text </b>

<ol>
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
</ol>
```

```
# Nadpis 1
## Nadpis 2

Takto vypadá **tučný text**.
```

1. První objekt v seznamu
2. Druhý objekt v seznamu
3. Třetí objekt v seznamu

Největší výhodou Markdownu je čitelnost i když není text vykreslen, to byl i hlavní cíl jazyka. Druhou hlavní výhodou jazyka je jednoduchost syntaxe, markdown obsahuje jen pár prostých znaků. Další výhodou je možnost vytvoření souboru v jakémkoliv zařízení s jakýmkoliv operačním systémem a jeho jednoduchá implementace. (9)

Markdown je možné najít například na stránkách GitHub i GitLab, StackOverFlow a Reddit.

3.2.3 CSS3

CSS neboli Cascade Style Sheets je textový dokument, který pomocí různých selektorů předává informaci o tom, jak se budou zobrazovat prvky HTML, například jakou barvu bude mít prvek, jaké formátování a kde bude umístěn. Číslo za zkratkou CSS znamená verzi kaskádového stylu, nejrozšířenější verze je verze 3. V CSS3 byly přidány funkce pro snadný vývoj responsivních aplikací umožňující změnu stylů vzhledem k velikosti obrazovky

zařízení a dále byly přidány animace, které je nyní možné psát jako součást kaskádových stylů. (6) (7)

Pro nastylování prvku je doporučeno vytvořit třídu *class* jako je v ukázce .error nebo *id #active*, pro které se definuje vzhled. Tento vzhled se ukáže u daného prvku ihned, nemusíme znovu načítat program. U přednastylovaných elementů, mohou být dané elementy přestylovány jako to je v případě body.

```
body {
  margin: 0;
  background-color: #800080;
}
#active {
  color: #800080;
}
.error{
 color: red;
}
```

Kaskádový styl si každý může postupně navrhnout sám anebo může použít framework, který má již předefinované třídy a styly. Nejznámější a nejpoužívanější framework je Bootstrap. Mezi další známe frameworky mohou být zařazeny Foundantion a PureCSS. (5)

SASS a SCSS

Při větším množství kódu CSS je velmi náročná a složitá údržba či změna kódu. Z toho důvodu vznikl SASS (Syntactically Awesome Style Sheets) pomáhá nám zbavit se rekurzivního kódu a udržuje náš CSS kód přehledný. SASS je předprocesor CSS, který navrhl Hampton Catlin a Natalie Weizenbaum, umožňuje nám používat funkce, které v CSS ještě neexistují, jako jsou proměnné, vnoření, mixiny, dědičnost a další. Aby kódu rozuměl i prohlížeč musíme celý kód zkompilovat a vygenerovat výstup v CSS. (7) (11)

Bude ukázán základní prvek SASS, proměnné. Proměnné mohou být použity v celé šabloně stylů a mohou být do nich uloženy například barva, styl písma nebo jakákoliv hodnota CSS. Zápis vypadá takto: (11)

```
$primary-color: #b529c9

  body
font: 100% $primary-color
```

Už u první ukázky kódu je možné si všimnout velkého rozdílu mezi SASS a CSS, nelze zde najít složené závorky ani středníky. SASS používá tzv. odsazenou syntaxi, odsazení

používá k oddělení bloku kódu a znaky k oddělení pravidel, a přípona souboru je .sass. (11) (12)

V roce 2006 vznikl předprocesor SCSS (Sassy Cascaded Style Sheets), který je velmi podobný CSS a vyplňuje mezery a nekompatibilitu mezi SASS a CSS. SCSS používá složené závorky k označení bloků kódu a středníky k oddělení pravidel, přípona souboru je .scss. (12)

```
$primary-color: #b529c9;

body{
font: 100% $primary-color;
}
```

Bootstrap

Bootstrap je jedním z nejpopulárnějších front-end frameworků a open-source projektů. Původně se nejednalo o open-source projekt, dříve známý jako Twitter Blueprint, cílem tohoto projektu byla konzistence zdrojového kódu. Projekt byl vytvořený malou skupinou designérů a vývojářů Twitteru, v čele této skupiny se nacházeli Mark Otto a Jacob Thornton. V polovině roku 2010 se rozhodli projekt vydat jako open-source s názvem Bootstrap, což znamená v doslovném překladu řemínek u boty. Projekt je uveřejněn na GitHubu, a je nadále je udržován Markem Ottou a Jacobem Thornronem a malou skupinou hlavních vývojářů, s pomocí velké skupiny přispěvatelů. (13) (14)

Pro použití Bootstrap je nutné používat HTML5 včetně meta informace viewport, která zajistí responzivní chování webu.

Zde se nachází ukázka kódu pro základní stránku Bootstrap verze 5.1.3, jedná se o šablonu stylů CSS a plugin pro JS.

```
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ
6jIW3" crossorigin="anonymous">

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.js"
integrity="sha384-
QJHtvGhmr9XOIpI6YVutG+2QOK9T+ZnN4kzFN1RtK3zEFEIsxhlmWl5/YE
SvpZ13" crossorigin="anonymous">
```

„Některé pluginy a komponenty CSS závisí na jiných pluginech.“ (13)

Bootstrap je možné přidat do projektu i pomocí balíčkovacího systému npm nebo yarn.

```
npm install bootstrap@5.2.0
yarn add bootstrap@v5.2.0
```

3.3 Design

Design je obecně považován za proces vytváření estetických a funkčních produktů nebo prostředí. Designéři jsou zodpovědní za zkoumání potřeb a preferencí uživatelů a navržení řešení, které splňují jak estetické, tak funkční požadavky.

V současné době je design stále důležitějším prvkem pro úspěch většiny společností. Estetický a funkční design přitahuje pozornost zákazníků a může pomoci zvýšit prodej a zisk.

Existují dva základní druhy webového designu, statický design a dynamický design.

3.3.1 Statický design

Webové stránky, u kterých je použit statický design jsou pevně dané a neměnné. Tyto stránky obsahují pouze fixní obsah a nereagují na změny uživatelského chování nebo zařízení, na kterém jsou zobrazovány.

Konkrétně se používá pro stránky, které poskytují informace o určité firmě nebo organizaci, nebo pro stránky, které slouží jako online vizitky nebo katalogy produktů. Tyto stránky obvykle nevyžadují žádnou interakci s uživatelem a jsou určeny pouze k prezentaci informací.

Statický design má řadu výhod, jako je například jednoduchost, rychlost načítání a nízké nároky na serverové prostředky. Na druhou stranu však neposkytuje uživatelům žádnou možnost interakce a neumožňuje aktualizaci obsahu webu bez zásahu webového vývojáře. (15) (16)

3.3.2 Dynamický design

Dynamický design se používá, jak lze již poznat dle názvu, pro stránky, které se mění v závislosti na uživatelském chování nebo zařízení, na kterém jsou zobrazovány. Tyto stránky jsou více interaktivní a mohou obsahovat živé aktualizace, animace a další prvky, které

reagují na uživatelské akce. Dynamický design je flexibilnější a přizpůsobivější, ale může být složitější na vývoj a údržbu.

Dynamický design umožňuje stránkám reagovat na uživatelské akce a zlepšuje uživatelský zážitek. Například, pokud uživatel klikne na tlačítko na stránkách, může se zobrazit nový obsah nebo se změnit vzhled stránky. Toto zajišťuje, že stránky jsou zábavnější a atraktivnější pro uživatele. Stránky mohou obsahovat živé aktualizace nebo online formuláře, které se aktualizují v reálném čase, což umožňuje uživatelům získat aktuální informace nebo provést akce bez nutnosti opakovaného načítání stránek.

Nevýhodou může být nižší rychlost načítání stránek, protože dynamický design obvykle vyžaduje větší množství kódu a dalších prvků, které se musí načíst, aby se stránky správně zobrazily a reagovaly na uživatelské akce. Další nevýhodou je nutnost implementace interaktivních prvků a skriptů, které reagují na uživatelské akce. To může znamenat, že je potřeba více času a úsilí na vývoj a testování stránek. (15) (16) (17)

Responsivní design

Responsivní design je moderní přístup k navrhování webových stránek, který zajišťuje, že stránka se zobrazuje správně na jakémkoli zařízení, ať už se jedná o počítač, tablet nebo mobilní telefon.

Tento design je velmi důležitý, protože v současné době většina lidí používá různá zařízení pro přístup k internetu a je třeba, aby se stránky správně zobrazovaly na všech z nich. (16) (17)

Responsivní design se skládá z několika technik, které se používají k vytvoření webových stránek. Tyto techniky zahrnují použití fluidních mřížek a proporcionálních jednotek, aby se obsah stránky přizpůsobil různým velikostem obrazovek, stejně funguje i použití media queries, kde je možné specifikovat konkrétní velikosti zařízení.

Největší a nejdůležitější výhodou responsivního designu je, že umožňuje stránkám správně fungovat na všech zařízeních, což zajišťuje lepší uživatelský zážitek a zvyšuje pravděpodobnost, že se uživatelé na stránky budou rádi vracet. Další výhodou je, že responsivní design umožňuje jednodušší správu a údržbu stránek, protože je potřeba vytvořit pouze jednu verzi stránek, která se poté přizpůsobí různým zařízením.

Nevýhodou designu je náročnější vývoj webových stránek, protože je nutné zajistit, aby se stránky správně zobrazovaly na různých zařízeních a velikostech obrazovek. Je tak potřeba více času a úsilí na vývoj a testování stránek. (16) (17)

Adaptivní design

Adaptivní design je velice podobný responsivnímu ale s rozdílem, že adaptivní se zaměřuje na přizpůsobení uživateli, a i jeho zařízení. Velmi často se adaptivnímu designu nesprávně říká responsivní. Avšak na rozdíl od responsivního se nepřizpůsobuje jenom velikost obrazovek, přizpůsobuje se také celý dynamický obsah webových stránek, takže je více personalizovaný. (16) (17)

Adaptivní design se zaměřuje na několik hlavních aspektů, které jsou důležité pro přizpůsobení uživatelského rozhraní:

- Flexibilita: Adaptivní design se snaží umožnit webovým stránkám a aplikacím, aby se přizpůsobily různým velikostem obrazovek a rozlišením. To se dělá pomocí technik jako je responzivní design, který umožňuje webovým stránkám reagovat na velikost obrazovky a přizpůsobit se jí, nebo pomocí technik jako je media query, které umožňují webovým stránkám přistupovat k informacím o zařízení a rozlišení obrazovky a přizpůsobit se jim.
- Přehlednost: Webové stránky a aplikace by měly být přehledné a srozumitelné pro uživatele. To se dělá pomocí dobře navrženého a strukturovaného rozvržení, které umožňuje uživatelům snadno najít to, co hledají. Obsah webové stránky by měl být rozdělen do logických sekcí a skupin.
- Přístupnost: Tento aspekt se zaměřuje na to, aby obsah webových stránek byl přístupný pro co nejširší spektrum uživatelů, včetně těch se zrakovým postižením nebo jinými omezeními. To znamená, že web by měl být navržen tak, aby byl snadno čitelný a přístupný pro všechny uživatele, bez ohledu na jejich omezení nebo schopnosti. (16) (17)

3.4 SEO

SEO, zkratka pro optimalizaci pro vyhledávače, je proces, jehož cílem je zlepšit viditelnost webové stránky v organických výsledcích vyhledávačů, a tím přivést relevantní návštěvníky na stránku a zvýšit návštěvnost a konverze. (18) (19)

„Pokud váš obsah není optimalizován pro vyhledávače, nezáleží na tom, jak dobrý je. Nikdo to nenajde.“ (20)

Klíčovými faktory ovlivňujícími pozici webové stránky v organických výsledcích vyhledávačů jsou klíčová slova, kvalita odkazů směřujících na stránku a technické SEO.

Je tedy důležité vytvořit relevantní obsah pro klíčová slova, která chtějí návštěvníci najít, a umístit je na stránku vhodným způsobem.

Dalším důležitým faktorem je kvalita odkazů, které směřují na webovou stránku. Vyhledávače berou v potaz počet i kvalitu odkazů, a tak je důležité budovat zpětné odkazy z relevantních a autoritativních zdrojů.

Technické SEO je také velmi důležité. Webová stránka by měla být rychlá, snadno prohledatelná pro vyhledávače a měla by být přizpůsobená pro mobilní zařízení. Důležité je také správné použití meta tagů, URL struktury a sitemapy. (18) (19)

3.5 Webové aplikace

Webové aplikace jsou aplikace, které fungují na webovém prohlížeči a jsou dostupné prostřednictvím internetu. Využívají se často k různým účelům, jako je například zpracování dat, zobrazování informací, vytváření obsahu nebo poskytování služeb online.

Webové aplikace bývají vyvíjeny pro stranu serveru a pro stranu klienta. Na straně serveru komponenty provádí zpracování dat a logiku aplikace, obvykle jsou použity jazyky jako je Node.js, PhP nebo Python. Na straně klienta se obvykle vyvíjejí pomocí jazyků jako JavaScript, HTML a CSS, které jsou používány pro tvorbu uživatelského rozhraní a interakci s uživatelem. Tyto aplikace mohou být hostovány na webovém serveru a přístupné pro uživatele prostřednictvím internetu.

Webové aplikace se obvykle dělí na dva hlavní typy: single-page aplikace (SPA) a multi-page aplikace (MPA). V této kapitole bude kladen důraz především na SPA, její vývoj a výhody či nevýhody při jejím použití. (7) (21) (22)

3.5.1 Multi-page applications (MPA)

Tradiční vícestránkové aplikace jsou statické webové stránky, které jsou rozděleny na více samostatných stránek a uživatel se musí přepínat mezi nimi, pokud potřebuje zobrazit různé informace nebo provést různé akce. MPA jsou obvykle využívány pro složitější aplikace, kde je potřeba zobrazit mnoho různých informací nebo provádět různé akce. Fungují přes server tzn., že pokud klient pošle dotaz musí počkat, než server odpoví, během toho nelze stránky procházet, „zamrznou“. MPA se tak stává velmi pomalé, protože přes server musí projít velký objem dat, jednotlivé stránky, a načíst se v klientovi. Z toho důvodu se vývojáři snažili najít jiný způsob návrhu webových stránek. (21) (22)

3.5.2 Single-page applications (SPA)

Single-page applications, v překladu jednostránková webová aplikace, je aplikace na straně klienta neboli prohlížeče a obsah aplikace je tvořen jedinou stránkou, která se načítá dynamicky. (7) (21) (22)

Na začátku SPA bylo hnutí AJAX, které dalo za vznik myšlence nového návrhu webových stránek. Zkratka AJAX je pojmenování pro Asynchronní JavaScript a XML. Asynchronní znamená že prohlížeč požaduje data z webového serveru, stránky nezamrznou a uživatel si je může prohlížet dál. Když server vrátí odpověď, provede změny na stránce, aniž by ji musel celou načíst znovu. SPA navazuje na AJAX a rozšiřuje techniky na úrovni stránek na celou aplikaci. Pro vývoj SPA jsou použity JavaScriptové frameworky, jako například Angular, React, Vue, Ember, Swelte a další. (21)

Výhody použití SPA

Největší výhodou je rychlost a doba odezvy. Jakmile bude jednou aplikace načtena, nebude se načítat při každém kliknutí. Tzn. rychlejší načítání a přívětivější uživatelské prostředí, jedná se o nejmodernější přístup pro vývoj softwaru. Rychlost načítání zajistí i menší odchodovost zákazníků ze stránky. (21) (22)

Pro vícestránkové aplikace (MPA) je často nutné přepsat celý kód pro zobrazení rozhraní, protože každá stránka má své vlastní rozhraní a většinou jsou tyto stránky mezi sebou propojené odkazy, tlačítky a dalšími prvky. To pro SPA neplatí, stačí upravit rozhraní zobrazování stránek pro určitý typ zařízení. (21) (22)

Jednou z dalších výhod je snadnější údržba kódu, pokud dojde k úpravě něčeho v záhlaví či zápatí stránky. Pro SPA stačí upravit jen málo souborů, abychom se dostali k požadovanému výsledku. Pro MPA to je ovšem jinak, musí se opravit každý soubor. Pro jednoduší údržbu kódu slouží architektonické vzory pro uspořádání soborů a složek v projektu jako je například MVC, MVP a MVVM. (21)

Nevýhody použití SPA

SPA jsou často velmi složité aplikace, které zpracovávají velké množství dat v reálném čase. To vyžaduje vysokou úroveň znalostí JavaScriptu, což může být pro méně zkušené vývojáře náročné.

Velké problémy jsou zaznamenány v SEO, důvodem je, že při pročtení první stránky se prohlížeč zastaví, protože nevidí žádné další hypertextové odkazy, které by následovaly.

Odešle stránku indexeru, který pak stránku vykreslí. Následně musí prohlížeč počkat až se indexer vrátí s dalšími stránky, které bude moct procházet.

Celý proces je velmi neefektivní a zdlouhavý. Prohlížeč nedokáže porozumět postavení stránek ihned ale až později, kdy je obsah načten ze strany klienta, web se tak neumístí mezi prvními pozicemi ve výsledcích vyhledávání, což zapříčiní nízkou návštěvnost stránek. (22)

3.6 Datové architektury

V této kapitole jsou popsány základy datových architektur, se zaměřením především na architekturu MVC. Další známé architektury jsou například MVP a MVVM, které jsou porovnány s architekturou MVC. (23)

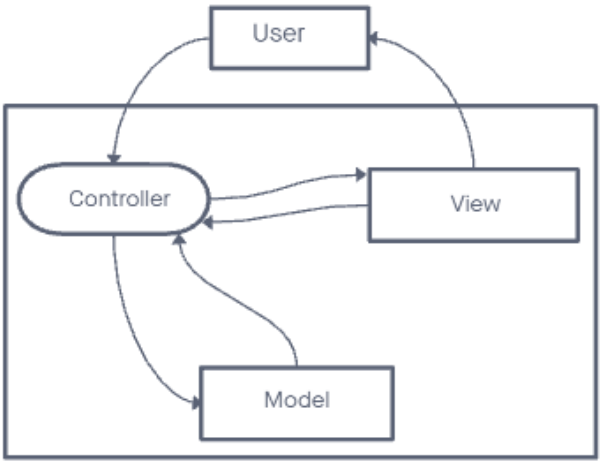
3.6.1 MVC

MVC neboli z angličtiny Model-View-Controller se skládá ze tří komponent, jak již plyne z názvu, které lze nezávisle na sebe upravovat. (23)

Model se stará o ukládání dat aplikace a za zpracování doménové logiky. Zajišťuje například vkládání zboží do košíku, korektní vyplnění formuláře a další činnosti. (23) (24)

View (pohled) poskytuje vizualizaci dat uložených v modelu a reprezentuje data v okně prohlížeče, nabízí i interakci s uživatelem. (23) (24)

Controller (řadič) zajišťuje komunikaci mezi modelem a view (pohledem), předává data. V žádném případě by controller neměl data zpracovávat. (24)



Obrázek 1 - Schéma MVC (24) (25)

Obrázek 7 ukazuje komunikaci v architektuře MVC. Příkladem komunikace může být hlasování na stránce. Uživatel odhlasuje, controller odešle data modelu a model data zpracuje.

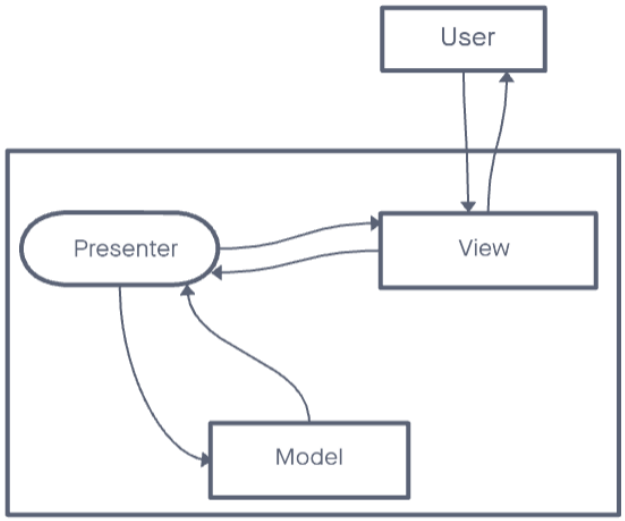
Následně vrátí finální data, která jsou controllerem předána do view, pohledu. View zobrazí tento výsledek uživateli.

Největší výhodou MVC je organizování obsáhlého kódu, což je zásluhou třívrstvé architektury, kód je tak snadno udržovatelný a jednodušší pro testování. Díky třívrstvé architektuře je aplikace flexibilní a dobře škálovatelná. Zároveň pomáhá urychlit proces vývoje, protože komponenty lze nezávisle na sebe upravovat. (23) (24) (25)

MVC funguje skvěle s JS, a i jeho frameworky, proto podporuje asynchronní volání metod, což umožňuje vytvářet rychlejší načítání webových aplikací. Ve spojení MVC a JS frameworky se jedná o ideální podmínky pro vznik SPA (Single-page aplikace), umožní to rychlý vývoj webových aplikací. (23)

3.6.2 MVP

Stejně jako MVC je MVP tří vrstvá architektura obsahující model, view (pohled), ale neobsahuje Conroller (řadič) ale presenter.



Obrázek 2 - Schéma MVP (23) (25)

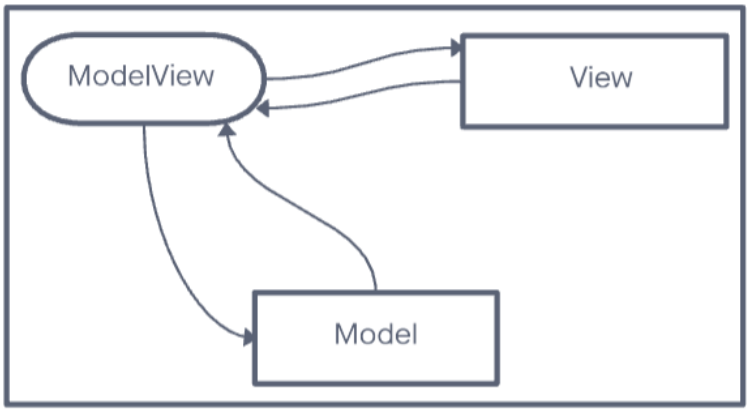
Model zastává stejnou funkci jako v architektuře MVC. Tento komponent zajišťuje, že data jsou správně uložena, zpracována a předávána mezi ostatními částmi systému. View také poskytuje vizualizaci dat uložených v modelu, ale také čeká na vstup uživatele, který zašle na vhodný presenter. Presenter zastává funkci controlleru, předává data mezi modelem a view (pohledem). Tento komponent zajišťuje, aby pohled zobrazoval aktuální stav dat a reagoval na změny v modelu. Presenter také zajišťuje, aby model zpracoval aktualizace dat vyplývající z uživatelských akcí zobrazených v pohledu. V MVP oproti MVC dostává veškeré požadavky view (pohled) a ten následně je odešle do presenteru. (23) (25)

Tento trojkomponentní model datové architektury je navržen tak, aby byl dostatečně jednoduchý a flexibilní pro vývoj MVP aplikace, a aby umožňoval uživatelům otestovat a poskytnout zpětnou vazbu na aplikaci v počáteční fázi jejího vývoje. (23) (25)

3.6.3 MVVM

MVVM (Model-View-ViewModel) je datová architektura, která se používá v některých platformách pro vývoj aplikací (např. WPF, Xamarin). Tato architektura se skládá ze tří hlavních komponentů: modelu, pohledu a viewmodelu.

Na rozdíl MVVM od MVC usiluje o oddělení logiky aplikace od uživatelského rozhraní. ViewModel pomáhá oddělit prezentační vrstvu a nabízí metody a příkazy pro správu stavu zobrazení a manipulaci s modelem. ViewModel také umožňuje snadnější testování a údržbu aplikace. (25) (26)



Obrázek 3 - Schéma MVVM (25) (26)

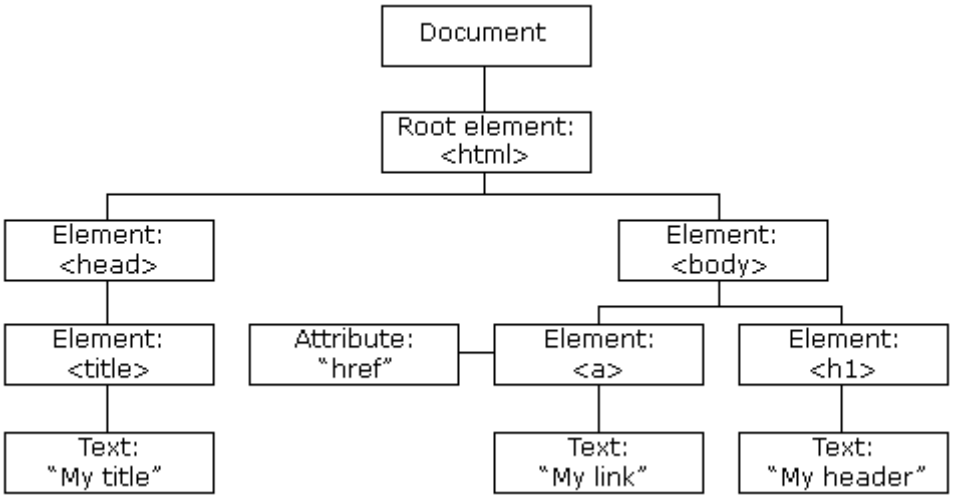
Zde je k vidění schéma MVVM, které vychází z návrhu MVC. Hlavním rozdílem je, že MVVM je převážně určen pro Silverlight a WPF aplikace, které slouží pro vývoj desktopových aplikací pro operační systémy Windows. (26)

3.7 DOM

Dokument Object Model je programovací rozhraní API pro dokumenty, jedná se o standard pro přístup k dokumentům, který stanovila W3C (World Wide Web Consortium).

Přistupovat k dokumentům je možné pomocí tří modelů:

1. Core DOM – pro všechny dokumenty
2. XML DOM – pro XML dokumenty
3. HTML DOM – pro HTML dokumenty



Obrázek 4 - HTML DOM – Document Object Model (27)

Document Object Model se vytvoří při načítání stránky, jedná se o strom objektů. Díky DOM může JavaScript vytvořit dynamické HTML. Programátoři tak mohou vytvářet a měnit dokumenty, procházet jejich strukturu a přidávat, upravovat nebo odstraňovat prvky a obsah.

3.7.1 Virtuální DOM

Virtuální DOM (Virtual DOM) je programovací koncept implementovaný knihovnami JavaScriptu, kde je virtuální prezentace uživatelského rozhraní uchována v paměti a synchronizována se skutečnou DOM knihovnou. Virtuální DOM má stejné vlastnosti jako skutečný DOM, ale postrádá sílu skutečné věci, tedy přímo změnit to co je na obrazovce. (28) (29)

Při přidání nových prvků do uživatelského rozhraní je vytvořen virtuální dokument objekt modelu, který je prezentován jako strom. Každý prvek je uzlem v tomto stromu, pokud se stav některých z těchto prvků změní, vytvoří se nový virtuální strom. Tento strom je pak porovnán s předchozím virtuálním stromem.

Jakmile je hotovo, virtuální DOM vypočítá nejlepší a nejrychlejší možnou metodu pro provedení těchto změn ve skutečném modelu. Tak je na skutečném modelu DOM provedeno minimální počet operací a snižují se tím nároky na výkon. (28) (29)

Virtuální DOM používají JavaScriptové frameworky, a to především React a Vue, o kterých tato diplomová práce pojednává.

3.7.2 Přírůstkový DOM

Přírůstkový DOM (Incremental DOM), který vyvinula společnost Google, je definován klíčovou myšlenkou:

> *„Každá komponenta je zkompilována do řady instrukcí. Tyto pokyny vytvářejí stromy modelu Document Object Model a aktualizují je na místě, když se data mění.“ (29)*

Přírůstkový DOM porovnává existující prvek v modulu DOM s prvkem, který je určen aktuálním voláním funkce, aby našel změny a aplikoval je na skutečný DOM. Narozdíl od virtuálního DOM, nevytváří žádný mezilehlý strom, ale strom je přímo na místě upravován.

Přírůstkový DOM nepotřebuje tak velkou paměť, protože ta je přidělena pouze pro změny a tím je v některých případech zvýšena rychlost, což je i hlavní výhoda přírůstkového Document Object Model. Používá se v JavaScriptovém frameworku Angular, který vyvinula společnost Google.

3.8 JavaScript

JavaScript je jeden z nejpopulárnějších interpretovaných kompilovaných programovacích jazyků. Je též známý jako skriptovací jazyk pro webové aplikace, který lze použít primárně na straně klienta tedy prohlížeče, tak ale i na straně serveru. Umožňuje vytvořit SPA, tedy nemusí se načítat celý obsah stránky pro každou akci. (30) (31)

> *„JavaScript má jedinečné postavení mezi programovacími jazyky, jelikož jako jediný se nachází na téměř všech osobních počítačích po celém světě“ (8)*

JS lze zapsat dvěma způsoby. Lze ho zapsat do dokumentu HMTL, přípona souboru .html, pod element <body>, kde se nachází element <script>, kam můžeme kód rovnou vepsat. Jak je možné si všimnout v značce script je atribut type, který konkretizuje, jaký skriptovací jazyk bude použit. Ačkoliv od HTML5 je JavaScript výchozím jazykem pro HTML5, a tak není potřeba přidávat type do scriptu v případě JavaScriptu. Nebo na něj lze v html kódu v element <script> odkázat na samostatný soubor, přípona .js.

1. Možnost
   ```
   <script type="text/javascript"> alert("Hello World!"); </script>
   ```

2. Možnost
   ```
   <script src="src/index.js"></script>
   ```

3.8.1 Historie JavaScript

V roce 1995 byl JS uveden Bredanem Eichem s názvem Mocha jako způsob přidávání programů na webové stránky ve společnosti Netscape, která o rok později vytvořila stejnojmenný prohlížeč Netscape Navigator podporující JS. Později se JS přejmenoval na LiveScript a nakonec na JavaScript. Název JavaScript, byl vybrán kvůli marketingovým účelům, nemá téměř nic společného s programovacím jazykem Java, za který je chybně velmi často zaměňován. (8) (30) (31)

Velkého úspěchu JavaScriptu si všimla i společnost Microsoft, která ihned vydala kompatibilní jazyk jménem JScript v kterém opravila i některé chyby JS. Následně i vydala podporu pro svůj prohlížeč Internet Explorer.

> *„Po jeho přijetí mimo Netscape byl napsán pro standardizaci dokument, který popisuje způsob, jakým by měl jazyk JavaScript fungovat, takže různé části softwaru, které tvrdily, že podporují JavaScript, ve skutečnosti mluvily o stejném jazyce." (31)*

Tato specifikace, které popisuje JavaScript se nazývá ECMAScript, který je normovaný neziskovou organizací ECMA International. (30) (31)

Během několika let se vyvinulo mnoho verzí JavaScriptu, nynější verze je verze ES11, avšak každý rok vychází nová aktualizace. Prohlížeče se tak každý rok musí aktualizovat kvůli podpoře JS na webových aplikacích. (31)

Verze	Rok	Příklady změn, které byly provedeny.
ES1	1997	První vydání JS
ES2	1998	Zajištění souladu specifikace s normou ISO/IEC 16262
ES3	1999	Přidání regulárních výrazů, nové ovládací příkazy.
ES4	-	Od této verze bylo opuštěno kvůli rozdílům ohledně jazykové složitosti. Některé ze změn jsou později zahrnuty v ES6.
ES5	2009	Důkladnější kontrola chyb v kódu, přidání getrů a setrů.
ES5.1	2011	Zajištění souladu specifikace s normou ISO/IEC 16262:2011
ES6	2015	Přidání deklarace tříd, iterátory a smyčky, výraz funkce šipky.
ES7	2016	Přidání blokování proměnných a funkcí, přidání operátoru umocňování.
ES8	2017	Přidání hodnoty objektu a funkce pro snadnou manipulaci s objekty.
ES9	2018	Přidání nových parametrů pro objekty a asynchronní iterátory.
ES10	2019	Změna návratových hodnot funkcí, vytvoření nových funkcí například Array.Flat() a Array.flatMap()
ES11	2020	Přidáno volitelné řetězení, vyhodnocení Null nebo nedefinováno.
ES12	2021	Přidání číselného oddělování, funkce .replaceAll() a Promise.any()

Tabulka 1 - Verze ECMA-262 (7) (32)

Výše v tabulce jsou nastíněné jednotlivé verze ECMA-262 včetně roku uveřejnění a některé změny, které byly provedeny.

3.8.2 Vanilla JavaScript

Název Vanilla JavaScript je termín, který se používá pro čistý JavaScript bez použití jakýchkoliv přidaných knihoven nebo frameworku.

Tento termín byl zaveden v roce 2012, kdy vznikly vtipné stránky s názvem *vanilla-js.com*. Vanilla JS jsou základní prvky JavaScript, které je třeba se nejprve se naučit před učením se různých frameworků. JavaScript používají nejznámější společnosti jako je například YouTube, Twitter, Amazon, Netflix a mnoho dalších. (33)

3.9 TypeScript

TypeScript je open-source jazyk vytvořený Microsoftem v roce 2012, který je silně typovaný, tzn. že všechny proměnné se musí nejprve deklarovat s jejich datovým typem. Existuje pět základních deklarací proměnných:

1. number – pro celá čísla i pro zlomky
2. string – pro znaky a řetězce
3. void – pro funkci, která nic nevrací
4. null – nevrátí se žádná hodnota nebo hodnota null
5. boolean – pro funkci, která vrátí true/false (pravda/lež)

Deklarace proměnných umožní lepší kontrolu kódu a zachytí chybu ihned v počátku, tedy není nutné kontrolovat celý kód od začátku, jak tomu bylo u JS. (34) (35)

Jedná se o nadstavbu jazyka JavaScript, neomezuje funkce jazyka JavaScript, ale přidává určité funkce, které byly dříve nedostupné jako je například abstraktní třídy, obecné typy, rozhraní, a další. Díky těmto přidaným funkcím musí TypeScript fungovat s kompilátorem. Kompilátor převede kód jednoho jazyka do druhého jazyka s tím, že zkontroluje správné použití datových typů. Pokud by kompilátor narazil na chybu, nedovolil by program ani spustit. (34) (35)

V prvním příkladu lze vidět TypeScript, kdy byla proměnné myNumber explicitně přiřazena datový typ number a poté byla inicializována hodnotou 10.

```
let myNumber: number;

myNumber = 10;

console.log(myNumber);
```

V druhém příkladu je proměnná "myNumber" definována bez specifikace typu. V JavaScriptu se totiž typ proměnné určuje až při přiřazení hodnoty.

```
let myNumber;

myNumber = 10;

console.log(myNumber);
```

Rozdíl mezi JS a TypeScriptem lze vidět nejen v prvním použití kódu, ale také především v příponě souboru, kdy přípona souboru bude .ts pro čistý TypeScript. Pokud bude použit TypeScript s podporou JSX elementů přípona souboru by byla .tsx.

TypeScript je díky svým vlastnostem velmi používaný a populární. Ačkoliv je v psaní kódu očekávána podobnost s JavaScriptem, bohužel tomu tak není, TypeScript je více podobný C# nebo Java. Dále je používán v jazyku React, Vue a Angular. (34) (35)

3.10 JavaScript Framework

Framework je doplňkové rozšíření, které má usnadnit vývojářům práci při tvorbě webových aplikací za použití funkcí a knihoven třetích stran. Tyto frameworky poskytují šablonu pro vývoj aplikací a nabízejí množství funkcí a nástrojů, které umožňují vývojářům vytvářet komplexní aplikace rychleji a efektivněji.

JS frameworků existuje velké množství a všechny mají rozdílná specifika, pomocí těchto specifik je důležité si vybrat framework, který bude mít potřebné vlastnosti pro zvolenou webovou aplikaci. Mezi populární JavaScript frameworky patří React, Angular a Vue.js Vybrané JS frameworky budou popsány v následujících kapitolách. (23) (36)

3.10.1 React

React je webový framework a také JS knihovna, který byl vyvinut skupinou vývojářů společně s Facebookem pro tvorbu uživatelských rozhraní. V průzkumu The State of JS byla tato knihovna za poslední tři roky hodnocena jako jedna z nejvíce používaných a populárních, což svědčí o jejím vysokém stupni obliby ve vývojářské komunitě. (37) (28)

React se velmi snadno učí a lze jej použít v jakémkoli typu projektu. React lze jednoduše propojit i s frameworkem Bootstrap. Pro vytvoření projektu s názvem my-app s propojením na Bootstrap, stačí spustit příkaz.

```
npx create-react-app my-app
cd my-app
npm start
```

Pro vytvoření projektu podporující TypeScript je potřeba zadat odlišný příkaz.

```
npx create-react-app my-app --template typescript
```

Pokud již byl projekt započat a vývojář se rozhodne TypeScript přidat později, může ho přidat do stávajícího projektu pomocí příkazu npm install --save-dev typescript .

Nicméně je potřeba dát tsc do části "skripty" v našem package.json.

```
"scripts": {
        "build": "tsc",
      // ...
},
```

Následně je potřeba vygenerovat pravidla ve speciálním souboru s názvem tsconfig.json pomocí příkazu npx tsc –init.

React je založen na opakovatelně použitelných komponentách, jedná se o logicky oddělené části kódu, které dávají dohromady úplnou aplikaci. Každá komponenta má vlastní stav a vlastnosti, což umožňuje snadné a efektivní řízení toho, co se zobrazuje na obrazovce. Komponenty mohou být tvořeny kódem v JavaScriptu nebo v JSX, což je syntaxe podobná HTML. (28) (37)

V Reactu lze pro navigaci mezi různými stránkami využít knihovnu React Router, která umožňuje snadné definování cest, jež mohou být specifikovány podle konkrétních URL adres. Definice těchto cest se provádí pomocí komponenty Route, která přijímá dva základní parametry – path a element. Parametr path určuje cestu k dané komponentě, zatímco parametr element určuje samotnou komponentu, která má být vykreslena při zadané cestě.

```
<Router>
    <Routes>
        <Route  path='/MiniGame' element={<MiniGame />}>
            {' '}
        </Route>
        <Route  path='/Gallery' element={<Gallery />}>
        </Route>

        <Route  path='/Resume' element={<Resume />}>
        </Route>
        <Route  path='/' element={<Home />}>
        </Route>
    </Routes>
</Router>
```

Obrázek 5 - React Router (vlastní zpracování)

Mezi výhody Reactu patří jeho vysoká úroveň abstrakce, což znamená, že tvůrci mohou rychle vytvářet a kombinovat složité uživatelské rozhraní. Umožňuje vytvářet efektivní a responzivní webové stránky, což je velmi důležité pro uživatelskou zkušenost. React také umožňuje snadné testování komponent a vyhledávání chyb, což pomáhá tvůrcům rychle odhalit a opravit problémy.

Hlavní nevýhodou Reactu je nekompletnost frameworku, takže je vývojář odkázán na nástroje třetích stran. Jednou z dalších výhod i nevýhod Reactu je možnost použití JSX s kombinací HTML, výhoda je ochrana kódu před SQL injection. Nevýhodou je, že JSX je trochu složitější na učení. (28) (38)

Na Reactu je vytvořeno plno známých webových aplikací jako je například Facebook, Instagram, Pinterest, Netflix, Airbnb, Reddit, Paypal a další. (28)

3.10.2 Vue

Vue je JavaScriptový framework pro vytváření UI, byl vydán v roce 2014 čínským programátorem Evan You. Navzdory tomu, že se jedná o relativně nejmladší framework, získal si popularitu díky tomu, že se dokázal poučit z chyb a nedostatků předchozích frameworků, a současně převzal některá výhodná řešení.

Je postaven na standardech jazyků HTML, CSS a JavaScript. Efektivně rozvíjí jednoduchá nebo složitá UI . Vue lze použít různými způsoby například pro vylepšení

statického kódu HTML, jako jednostránkovou aplikaci SPA, Fullstack (vykreslování na straně serveru), Jamstack (generování statických stránek) a další.

Vue poskytuje velké množství pluginů a rozšíření, které umožňují rozšířit jeho funkce a přizpůsobit ho konkrétním potřebám. Tyto pluginy mohou být použity pro různé účely, jako je například práce s animacemi, integrace s různými knihovnami a rozšíření frameworku o další funkce.

Pro vytvoření jednoduchého projektu stačí zadat příkaz npm init vue@latest, který nainstaluje poslední dostupnou verzi Vue. (39) (40)

Tento příkaz spustí oficiální nástroj pro generování uživatelského rozhraní projektu Vue. V nástroji si následně lze vybrat, jaké funkce bude obsahovat projekt jako je například TypeScript, router pro Single-page aplikace a další. (39)

```
✓ Project name: … my-app
✓ Add TypeScript? … No / Yes
✓ Add JSX Support? … No / Yes
✓ Add Vue Router for Single Page Application development? … No / Yes
✓ Add Pinia for state management? … No / Yes
✓ Add Vitest for Unit testing? … No / Yes
✓ Add Cypress for both Unit and End-to-End testing? … No / Yes
✓ Add ESLint for code quality? … No / Yes
✓ Add Prettier for code formatting? … No / Yes

Scaffolding project in ./my-app
    Done.
```

Bootstrap je možné do projektu přidat úplně stejně jako u frameworku React pomocí balíčkovacího systému. Stejně jako React, Vue využívá komponentový model, který usnadňuje použití často opakovaných částí aplikace, například zápatí nebo navigační lišty.

Pro vytvoření SPA aplikace je nutné využít Vue Router, který je potřeba nejprve nainstalovat pomocí správce balíčků npm. Vue Router umožňuje definovat různé cesty (routes) a přiřadit k nim jednotlivé komponenty, které se budou zobrazovat při přechodu mezi jednotlivými částmi aplikace. Poté již lze vytvářet routy pomocí komponenty Router a jejích vnořených komponent Route. (39) (40)

```
import { createRouter, createWebHistory } from "vue-router";
import Home from "../views/HomeView.vue";
import Gallery from "../views/GalleryView.vue";
import MiniGame from "../views/MiniGameView.vue";
import Resume from "../views/ResumeView.vue";

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: "/",
      name: "Home",
      component: Home,
    },
    {
      path: "/gallery",
      name: "Gallery",
      component: Gallery,
    },
    {
      path: "/minigame",
      name: "MiniGame",
      component: MiniGame,
    },
    {
      path: "/resume",
      name: "Resume",
      component: Resume,
    },
  ],
});

export default router;
```

Obrázek 6 - Vue Router (vlastní zpracování)

Hlavní výhodou Vue je založení na Vite, což je oficiální nástroj pro sestavení Frontendu, který je moderní, jednoduchý a extrémně rychlý. V porovnání s jinými frameworky je Vue velmi snadno pochopitelný a jeho syntaxe je intuitivní. To znamená, že i začátečníci se mohou rychle naučit pracovat s Vue a vytvářet s ním kvalitní UI.

Další výhodou Vue je jeho schopnost vykreslovat obsah na straně serveru, což je vhodné pro optimalizaci SEO, tedy pro indexaci stránek vyhledávači. Toto řešení může pomoci umístění stránek na vyšší pozice ve výsledcích vyhledávání. (39)

Vue stejně jako React využívá virtuální DOM, který umožňuje obnovovat pouze ty části stránky, ve kterých došlo aktivitou uživatele ke změně. Nemusí se tak načítat celý obsah webové stránky znovu a znovu. (39) (40)

Jeho hlavní nevýhoda je malá komunita podpory oproti React nebo Angular frameworku. (39)

3.10.3 Angular

Angular je framework pro návrh aplikací a vývojová platforma pro vytváření efektivních jednostránkových aplikací. Byl vyvinut společností Google a často bývá porovnáván s Reactem. (36) (41) (42)

Angular má i svého předchůdce AngularJS, obě verze mají odlišnou syntaxi i princip fungování, a tak nejsou kompatibilní. AngularJS byl vydán v roce 2010 a využívá JavaScript, jak vyplývá z názvu. Zatímco Angular využívá TypeScript a byl vydán v roce 2016. (41)

Angular framework je velmi úzce spjat s node.js, který je potřeba mít nainstalovaný. Node.js lze nainstalovat z oficiálních stránek: https://nodejs.org/en/download/ (41)

Pro vytvoření jednoduchého projektu stačí zadat příkaz, který nainstaluje poslední dostupnou verzi Angular.

```
ng new my-app
```

„ng new nainstaluje potřebné balíčky npm a další závislosti, které Angular vyžaduje. To může trvat několik minut.“ (41)

```
? Would you like to add Angular routing? Yes/No
? Which stylesheet format would you like to use?
  >CSS
  >SCSS  [ https://sass-lang.com/documentation/syntax#scss]
  >Sass    [ https://sass-lang.com/documentation/syntax#the-indented-syntax]
  >Less    [ http://lesscss.org]

✓ Packages installed successfully.
```

Stejně jako React a Vue používá i Angular komponenty jako stavební bloky, které tvoří aplikaci. Komponenty lze znovu použít v různých částech aplikace a lze je taky snadno od sebe oddělit, a tak můžou být nahrazeny lepšími verzemi. (36) (41)

Pro vytvoření komponenty je, ale zapotřebí na rozdíl od React a Vue, zadání příkazu do příkazové řádky, což může být považováno za nevýhodu. Po spuštění příkazu vytvoří Angular složku pojmenovanou po vytvořené komponentě.

```
ng generate component my-component
```

Komponenta se vytvoří jako složka, která obsahuje několik souborů. Konkrétně se jedná o soubory CSS, HTML, TypeScript a testovací soubor komponenty. (41)

V rámci frameworku Angular je poskytován modul pro navigaci mezi stránkami, který je vyvíjen a spravován týmem Angular core. Tento modul poskytuje kompletní knihovnu pro navigaci a nabízí různé strategie porovnávání cest, intuitivní přístup k parametrům a zabezpečení proti neoprávněnému přístupu. Definice jednotlivých cest v rámci modulu Angular Routing probíhá stejným způsobem jako u předešlých verzí frameworku. Skupina samostatných cest je reprezentována objekty, které obsahují parametr "path" odkazující na část URL adresy, která určuje zobrazení a která je asociována s konkrétní komponentou. Na základě poskytnuté cesty Router provádí navigaci na požadovaný obsah. (35) (41)

Jak již bylo zmíněno Angular je psán pomocí jazyka TypeScript, což má značnou výhodu, protože ten eliminuje běžné chyby při psaní kódu, díky statickému typování. (35)

Další výhodou Angular je jeho čitelnost. Pokud bude do projektu přidán nový vývojář, díky zapouzdření se může lépe orientovat v kódu, a bude rychleji schopen produktivně pracovat na kódu.

Na rozdíl od React a Vue, Angular nepoužívá virtuální DOM, ale svůj vlastní mechanismus pro detekci změn v kódu, tzv. přírůstkový DOM. (41)

Využití Angularu je možné najít na stránkách Alza, Google, Weather.com, PayPal, ale také v různých interních aplikacích firem jako je například IBM nebo Cisco.

3.11 Teoretické porovnání JS Frameworků

Vue, Angular i React jsou založeny na komponentách a umožňují rychlé vytváření funkcí uživatelského rozhraní, avšak všechny mají jinou strukturu a architekturu.

Věc, která je velmi příznivá pro Vue je rychlost učení. Většina projektů má kořenovou složku app.vue a řadu podřízených komponent pro zobrazení různých věcí, které v sobě uchovávají veškerý front-end kód. (40)

Angular pro vytvoření komponenty musí znát příkaz a následně vytvoří soubor komponenty, kde se ukrývá veškerý potřebný kód. Angular má spoustu konceptů a syntaxí,

kterou je potřeba se naučit pro plné ovládání tohoto frameworku. Nejlepšího výsledku u Angularu lze dosáhnout s TypeScriptem, který je potřeba se též naučit. (41)

React je na tom o něco lépe než Angular, protože veškerý kód je obsažený v komponentě. Za velkou nevýhodu při učení u Reactu je učení JSX, které je s Reactem spjato. (28)

3.11.1 Výsledky ankety The State of JS

The State of JS je anketa, která každý rok shromažďuje data od uživatelů a následně je vyhodnocuje. Díky ní lze získat časovou osu používání jednotlivých frameworků a jak byly hodnoceny.

V roce 2019 se zúčastnilo průzkumu 21 717 respondentů. Z toho nejvíce respondentů odpovídalo z těchto zemí: 23.7 % z USA, 6.4 % z Velké Británie, 5.5 % z Francie, 5.1 % z Německa, 4.0 % z Kanady. Většinou se jednalo o respondenty, kteří mají dlouhé roky zkušeností s JavaScriptem a CSS. Když byli dotázáni na znalosti ohledně Backendu, 11.5 % buď nikdy nepracovali na Backendu nebo jsou začátečníci. Většina dotazovaných má negativní názor na Angular, pozitivní na Vue, a nejpozitivnější odezvu na React. (37)

V roce 2020 se zúčastnilo průzkumu 23 765 respondentů. Z toho nejvíce respondentů odpovídalo z těchto zemí: 21.1 % z USA, 6.0 % z Francie, 5.9 % z Německa, 5.8 % Velká Británie, 3.8 % z Ruska. Většinou se jednalo o respondenty, kteří mají dlouhé roky zkušeností s JavaScriptem a CSS. Když byli dotázáni na znalosti ohledně Backendu, 12.5 % buď nikdy nepracovali na Backendu nebo jsou začátečníci. Dotazovaní mají stále negativní smýšlení o Angularu, přesto je daleko používanější než Vue. React je nejpoužívanějším a nejlépe hodnoceným frameworkem a také má největší komunitu. (43)

V roce 2021 se zúčastnilo průzkumu 16 085 respondentů. Z toho nejvíce respondentů odpovídalo z těchto zemí: 14.0 % z USA, 4.8 % z Německa, 4.5 % z Ruska, 4.2 % z Francie. Většinou se jednalo o respondenty, kteří mají dlouhé roky zkušeností s JavaScriptem a CSS. Dotazovaní začínají Vue používat častěji, než tomu bylo minulý rok, jak vyplývá z ankety a odcházejí od Angular. React je stále nejvíc v podvědomí respondentů, nejpoužívanějším a nejlépe hodnoceným frameworkem a také má největší komunitu. (44)

3.11.2 Metriky front-end frameworků

Při výběru frameworků je třeba myslet i na výkon, pokud je projekt malý o výkon se není potřeba tolik starat. Avšak čím víc projekt poroste do rozsahu a složitostí, tím se výkon může stát větším problémem. Proto je dobré sledovat metriky výkonu a rozdíly mezi nimi.

Všechna níže uvedená měřítka pocházejí z analýzy Stefana Krauseho, který ji publikuje na stránce Github . Obrázek ukazuje nejaktuálnější benchmark test, který byl proveden na MaBook Pro 14 (32 GB RAM, 8/14 Cores, OSX 12.6), Chrome 107.0.5304.62 (arm64)) v listopadu v roce 2022. (45)

První tabulka ukazuje, jak všechny frameworky fungují ve většině srovnávacích testů, jako je vytváření nebo připojování řádků v tabulce. Vue je pomalejší než Angular ale zároveň výrazně rychlejší než React při výběru řádků. Na druhou stranu, Angular a React nejsou příliš efektivní při výměně řádků.

Name Duration for...	vue-v3.2.37	angular-v13.0.0	react-v17.0.2
Implementation notes			
create rows creating 1,000 rows (5 warmup runs).	46.2 ± 0.5 (1.21)	47.8 ± 0.5 (1.25)	52.6 ± 0.4 (1.38)
replace all rows updating all 1,000 rows (5 warmup runs).	46.9 ± 0.7 (1.16)	51.7 ± 0.7 (1.28)	53.9 ± 0.8 (1.34)
partial update updating every 10th row for 1,000 rows (3 warmup runs). 16x CPU slowdown.	121.9 ± 2.1 (1.16)	115.0 ± 2.8 (1.10)	145.8 ± 4.7 (1.39)
select row highlighting a selected row. (5 warmup runs). 16x CPU slowdown.	22.2 ± 1.2 (2.00)	16.4 ± 1.3 (1.48)	42.8 ± 0.9 (3.87)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4x CPU slowdown.	32.3 ± 1.0 (1.12)	180.3 ± 1.1 (6.25)	175.2 ± 1.4 (6.08)

Obrázek 7 - Benchmark test Vue, React, Angular 1/2 (45)

remove row removing one row. (5 warmup runs). 4x CPU slowdown.	53.8 ± 0.6 (1.14)	50.3 ± 1.7 (1.06)	58.0 ± 1.1 (1.23)
create many rows creating 10,000 rows. (5 warmup runs with 1k rows).	492.0 ± 2.1 (1.18)	512.3 ± 2.6 (1.23)	691.2 ± 3.7 (1.66)
append rows to large table appending 1,000 to a table of 10,000 rows. 2x CPU slowdown.	99.7 ± 0.4 (1.13)	109.4 ± 0.7 (1.24)	122.8 ± 0.4 (1.39)
clear rows clearing a table with 1,000 rows. 8x CPU slowdown. (5 warmup runs).	40.8 ± 1.3 (1.32)	72.6 ± 1.5 (2.35)	45.4 ± 1.6 (1.47)
geometric mean of all factors in the table	1.25	1.58	1.85

Obrázek 8 - Benchmark test Vue, React, Angular 2/2 (45)

Dle geometrického průměrů všech údajů v tabulce se Vue řadí na první příčku v rychlosti, a to díky výběru, výměně a vytvoření řádku, kdy je Vue opravdu rychlejší než Angular, který je rychlejší než React.

Name	vue-v3.2.37	angular-v13.0.0	react-v17.0.2
consistently interactive a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	2,101.5 ± 0.1 (1.17)	2,779.5 ± 0.6 (1.54)	2,551.7 ± 49.4 (1.42)
total kilobyte weight network transfer cost (post-compression) of all the resources loaded into the page.	196.5 ± 0.0 (1.37)	291.4 ± 0.0 (2.04)	274.6 ± 0.0 (1.92)
geometric mean of all factors in the table	1.27	1.77	1.65

Obrázek 9 - Doba spuštění (45)

Vue má i nejlepší metriky spuštění, díky velmi malé velikosti tohoto frameworku což pomáhá výrazně zkrátit dobu spuštění. Velmi podobné výsledky vidíme i pro React. Nicméně, Angular trpí svou těžší strukturou.

Name	vue-v3.2.37	angular-v13.0.0	react-v17.0.2
ready memory Memory usage after page load.	0.9 (1.41)	1.6 (2.53)	1.1 (1.69)
run memory Memory usage after adding 1,000 rows.	3.7 (2.10)	4.7 (2.66)	4.9 (2.79)
update every 10th row for 1k rows (5 cycles) Memory usage after clicking update every 10th row 5 times	3.7 (2.08)	4.8 (2.67)	5.4 (3.05)
creating/clearing 1k rows (5 cycles) Memory usage after creating and clearing 1000 rows 5 times	1.2 (1.78)	2.3 (3.36)	1.8 (2.62)
run memory 10k Memory usage after adding 10,000 rows.	26.7 (2.41)	28.8 (2.59)	35.7 (3.22)
geometric mean of all factors in the table	1.92	2.75	2.61

Obrázek 10 - Doba alokace paměti (45)

Na první pohled je patrné, že Angular je pomalejší na rozdíl od Vue a React. “Skutečný" DOM a obousměrný proces datové vazby způsobují, že výkon Angular je pomalejší než React.

3.12 Node.js

Node.js je open-source multi-platformový serverový framework jazyka JavaScript, který je velmi často spojován s tvorbou SPA nebo API. Navržen je pro vytváření škálovatelných síťových aplikací, přičemž slovem škálovatelných se rozumí schopných obsloužit mnoho připojených klientů naráz. Jádrem technologie Node.js je V8 JS (ES6). (46) (47)

Nejnovější verzí Node.js, kterou je možné si nainstalovat je verze 18.10.00. (48)

3.12.1 Architektura Node.js

Jádro celé architektury tvoří model "Single Threaded Event Loop", který slouží ke správě několika souběžných klientů. Jedná se o model založený na událostech jazyka

JavaScript a mechanismu zpětného volání jazyka. U architektury je možné se setkat s těmito pojmy:

- Asynchronní model – umožňuje provádět operace bez nutnosti čekat na dokončení předchozích operací
- Neblokování vstupně/výstupních operací – Neblokující operace znamená, že se server nezablokuje pro jeden požadavek.

Komponenty architektury

- Requests (požadavky).
  V závislosti na akcích, které musí uživatel provést, mohou být požadavky na server blokující (složité) nebo neblokující (jednoduché).

- Node.js Server.
  Server Node.js zpracovává příchozí požadavky od uživatelů, provádí potřebné operace a následně vrací výsledky zpět uživatelům.

- Event Loop (smyčka událostí).
  Event Loop je prostor, který přijímá veškeré uživatelské požadavky jako události a přiděluje jednotlivým nezávislým zdrojům.

- External Resources (externí zdroje).
  Pro zpracování blokujících požadavků od klientů v Node.js se často využívají externích zdrojů. Tyto externí zdroje mohou být různého typu, například výpočetní, skladovací a podobně.

Jedná se o škálovatelnou síť, tzn. že se vše řeší efektivně a lze zpracovávat více požadavků a událostí naráz. Pomocí kódu v jazyce Node.js je ukázáno na Obrázek 11, jak může být Node.js použit pro tvorbu webových aplikací a jak využít moduly a externí knihovny pro vytvoření škálovatelné aplikace. (46) (47)

```
1   //system constant
2   const fs = require("node:fs/promises");
3   const path = require("node:path");
4
5   // thirdparties constat
6   const express= require("express");
7   const app = express();
8   const cors = require("cors");
9   app.use(cors());
10
11  //NOTchange constant
12  const PORT = 8088;
13  const ALBUMS_ROOT_PATH = "/albums"
14
15  app.get(`${ALBUMS_ROOT_PATH}/:slug`, (req, res) => {
16      res.send("ahoj");
17    });
18
19  app.listen(PORT,()=>console.log(`Server lintening on ${PORT}`));
```

Obrázek 11 - Ukázka kódu v Node.js, (vlastní zpracování)

Začátek kódu ukazuje, jak jsou moduly zahrnuty do aplikace. Použity jsou `fs` a `path`, a jako systémové konstanty by se měly nacházet vždy na začátku kódu, protože umožňují pracovat se souborovým systémem. Cestu je možné upřesnit díky „ node: “, avšak lze psát i bez upřesnění cesty.

Dále jsou použity externí moduly, jako je `express` a `cors`, jedná se o doplňky třetích stran, které jsou propsány do konstant. První použitý externí modul je Express. Jedná se o open-source framework umožňující snadno vytvářet a spustit webové aplikace ve webovém prohlížeči, pomocí něho je vytvoření API velmi jednoduché a intuitivní. Další použitý doplněk je Node.js modul `cors`, který umožnuje nastavit Cross-Origin Resource Sharing (CORS), což je mechanismus zabezpečení, který umožňuje určit jaké domény a servery budou mít přístup. (46) (47)

Následuje kód samotné aplikace. Nejprve jsou uvedeny konstanty a proměnné, v našem příkladě se jedná o port, na kterém bude aplikace naslouchat, a o kořenovou cestu pojmenovanou jako `/albums`. (48)

Objekt app se pojí s metodou GET, app.get(). Metoda GET předává dva parametry, , prvním parametrem je cesta „ ${ALBUMS_ROOT_PATH}/ :slug[1] “ a druhým je funkce

[1] :slug znázorňuje jakýkoliv napsaný text psaný.

callback, která přebírá požadavek a odpověď na něj, „ req “ a „ res “. V tomto příkladě bude na stránku http://localhost:8088/albums/something odpověď „ ahoj“.

Pro poslouchání aplikace na konkrétním portu se používá metoda LISTEN, která obsahuje dva parametry, první parametr je port a druhým parametrem je callback, tedy funkce zpětného volání, která se zavolá až aplikace začne poslouchat na portu. (48)

3.12.2 Výhody a Nevýhody Node.js

Výhody:

+ Vhodné řešení pro real-time apps, které musí fungovat s velkým množstvím požadavků.
+ Využívá JavaScript, který spadá do základních dovedností každého programátora.
+ Díky JS je tak jednodušší přecházet mezi Frontendem a Backendem.

Nevýhody:

– API se velmi často mění a není zpětně kompatibilní, tím se stává API nestabilní.
– Nevhodný pro výpočetně náročné aplikace, dokáže obsloužit složitější aplikace ale nikoliv náročnější. (46) (47) (48)

3.13 Zprovoznění aplikace

Před zprovozněním aplikace je nutné zkontrolovat, jak se stránka zobrazuje ve více prohlížečích. Obzvláště ve starších verzích prohlížečů by mohlo dojít k problému.

Stránky mohou být následně převedeny na server, který bude poskytovat hosting stránkám a tím je zpřístupní široké veřejnosti. (6)

3.13.1 Doména

„Než návštěvníci uvidí váš web, musíte k němu přiřadit název domény. Můžete si zaregistrovat svůj vlastní název domény a poté najít webového hostitele, který bude vaši stránku obsluhovat komukoli, kdo navštíví doménu v prohlížeči.“ (6)

Doména slouží jako jedinečný identifikátor webové stránky a umožňuje uživatelům snadno najít a přistupovat k těmto stránkám prostřednictvím internetu. Domény jsou

registrovány a spravovány organizací ICANN (Internet Corporation for Assigned Names and Numbers), která zajišťuje jedinečnost a unikátnost domén po celém světě.

Doména neboli doménové jméno stojí v řádu několika stovek korun českých na rok, pokud by byla využívána národní doména .cz nebo evropskou doménu .eu. U ostatních domén je nutné si připlatit ještě pár stovek korun navíc.

Pokud má uživatel zájem o generickou doménu, nespojenou s konkrétním státem, je nutné si připlatit až několik tisíc korun českých za rok. (6)

3.13.2 Hostitel

Webový hostitel anglicky web hosting je poskytovatel internetových služeb nebo společnost, která poskytuje web hostingové služby. Většinou prodává nebo pronajímá paměťový prostor na svých serverech, který zajišťuje webu jedinečnou internetovou adresu. Tuto adresu je možné zadat do adresního řádku nebo vyhledat pomocí vyhledávače. (6)

Existuje několik druhů web hostingu. Každý typ hostingu je určen pro konkrétní případy užití, které ovlivňují cenu.

- Dedikovaný – plná kontrola nad serverem lepší úroveň zabezpečení a vyšší výkon, ceny u tohoto hostingu jsou ty nejvyšší od $ 80 - $ 500 za měsíc. U dedikovaného webhostingu existují 2 typy
    - spravovaný–nabízí profesionální podporu
    - nespravované–je nutné mít pokročilé technické znalosti a dovednosti
- Sdílený
    - Zdarma – většinou jsou tyto webhostingy bez zákaznické podpory, což je velká nevýhoda, obzvlášť ve chvíli, kdy je potřeba pomoct
    - Placený – používá se pro menší podniky které nemají velké objemy dat, obvykle za rozumnější cenu, $ 2 - $ 15 / měsíc, což z něj činí nejdostupnější možnost.
- Cloud hosting – Používá více virtuálních serverů k hostování webových stránek. Odolnější vůči problémům s fyzickým hardwarem, protože využívá síť vzdálených serverů. Cena je okolo od $ 10 - $ 200 / měsíc.
- Hosting virtuálního privátního serveru (VPS) - Poskytovatel hostingu poskytne plný root přístup k instalaci vlastního softwaru a operačního systému. Cena se pohybuje od $ 20 - $ 100 / měsíc. (6)

3.13.3 Přesun aplikace na server

Weboví hostitelé většinou nabízejí odlišné možnosti publikování stránek pomocí FTP (FTPS), pomocí grafického webového rozhraní nebo přímého nakopírování na server, to lze ale jen ve speciálních případech. (6) (49)

Nejčastěji se publikace webových stránek provádí přes FTP, což je File Transport Protokol. Jedná se o protokol, který přenáší soubory z počítače na úložiště poskytovatele. Zkratka FTPS znamená zabezpečenou FTP komunikaci, tedy protokol využívá pro přenos šifrování SSL/TLS. (6)

Pro přístup do FTP je potřeba znát heslo, uživatelské jméno a FTP adresu svého serveru. Mezi nejznámější FTP klienty patří Total Commander, Filezilla, WinSCP, FreeCommander. Všichni tito klienti umí pracovat jak šifrovaně, tak i bez šifrovacího protokolu. (49)

> *„Uživatelské rozhraní, které usnadňuje uživateli práci s programy prostřednictvím grafické prezentace určitých činností pomocí oken, dialogových rámečků, ikon, menu a dalších grafických prvků; uživatel nemusí nutně znát příkazy pro komunikaci s počítačem a jejich syntaxi.“ (50)*

Grafické webové rozhraní je velmi intuitivní na ovládání a nachází se jenom na pár web hostingách. Velkou nevýhodou grafického webového rozhraní je že po přihlášení se uživatel nachází na úpravě celého webu. (49)

4 Vlastní práce

Empirická část práce zpracovává témata, která souvisí s vývojem moderních JavaScriptových aplikací. Zaměřuje se na využití nejnovějších technologií a postupů pro vývoj aplikací pomocí JavaScript Frameworků s TypeScriptem, který slouží k vytvoření interaktivních prvků a funkcí na stránce. Konkrétně se jedná o JavaScriptové frameworky Vue, Angular a React.

Pro vývoj webové stránky byly využity teoretické znalosti získané v rámci práce, které byly aplikovány při vytváření informační architektury a wireframu. Tyto nástroje pomáhají orientovat se v umístění jednotlivých prvků na stránce a zajistit tak její přehlednost a uživatelskou přívětivost. Použity byly principy adaptivního designu, což znamená, že tímto způsobem může být stránka přístupná na všech zařízeních, ať už se jedná o počítače, tablety nebo chytré telefony. Zároveň jsou také dodržena pravidla přístupnosti, aby si i lidé s omezeními mohli prohlížet webové stránky.

Všechny weby se dorozumívají se serverem, který poskytuje potřebné údaje o galerii a textu webových stránek v Markdown souborech. Následně si jednotlivé JavaScriptové frameworky vytvořily komponenty, které jim pomáhají při komunikaci se soubory na serveru a generují, jak text vytvořený v Markdownu, tak vygenerované obrázky.

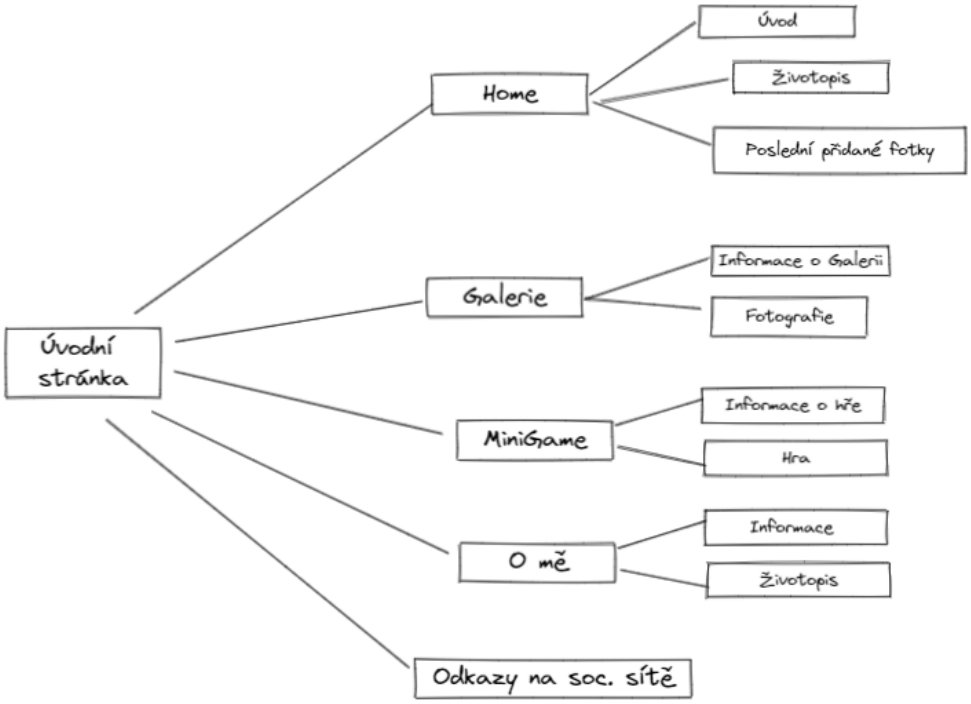
Ve všech komponentách byly využity knihovny třetích stran jako například v Markdownu doplněk marked, který pomáhá při kompilaci Markdownu. Všechny doplňky třetích stran jsou ve stejných verzích, aby nedocházelo k neočekávanému chování v aplikaci.

4.1 Uživatelské a grafické rozhraní

Pro uživatelské a grafické rozhraní byla vytvořena Informační architektura, Wireframe, pomocí kterého byly navrženy struktury a layoutu webové stránky. Díky těmto nástrojům byly rozebrány jednotlivé části webové stránky a navrženo jejich umístění tak, aby vytvořily přehlednou a intuitivní strukturu pro uživatele.

4.1.1 Informační architektura

Informační architektura je důležitou součástí tvorby webu, zabývá se organizací obsahu a umožňuje uživatelům rychleji pochopit, jaký obsah bude na stránce zobrazen, a tak se dostat k požadovanému cíli.



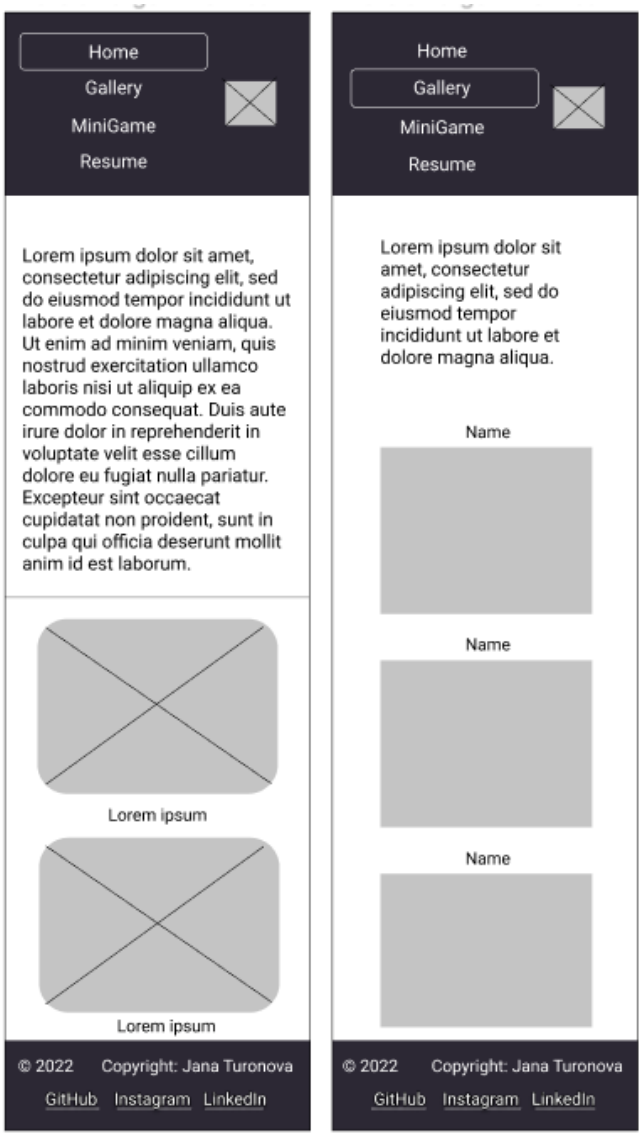
Obrázek 12 - Informační architektura (vlastní zpracování)

4.1.2 Wireframe

Ve druhém kroku vývoje webových stránek byl vytvořen náhled stránky, což je černobílý nákres, který jednoznačně určuje strukturu rozložení prvků a charakterizuje jednotlivé sekce stránky. Tento náhled pomohl lépe pochopit, jak bude stránka vypadat a jak budou jednotlivé prvky zde umístěny.

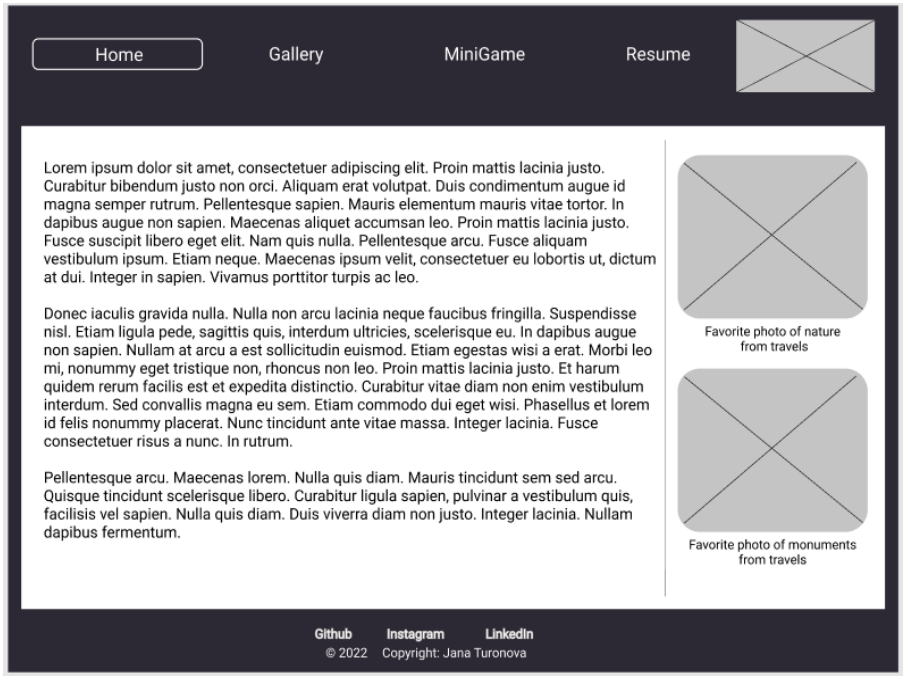
Webová stránka je navržena jako "mobile first", což znamená, že byla vytvořena nejprve pro menší obrazovky a následně pro větší zobrazení. Tento přístup je v současné době velmi

populární, z důvodu rozšiřujícího se používání chytrých telefonů a jejich přístupu k internetovému připojení a je třeba zajistit, aby webové stránky byly přístupné i na těchto zařízeních.



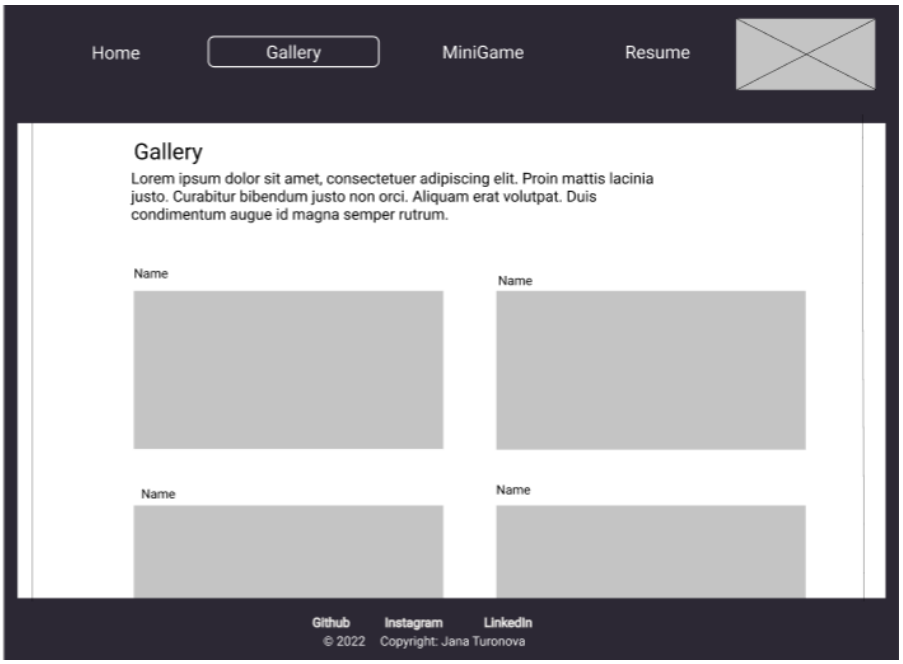
Obrázek 13 - Návrh webové stránky pro mobilní zařízení (vlastní zpracování)

První wireframe zobrazuje náhled úvodní obrazovky, která slouží k seznámení návštěvníka s obsahem stránky, zde se nachází dva obrázky, které motivují návštěvníka k návštěvě stránky galerie.

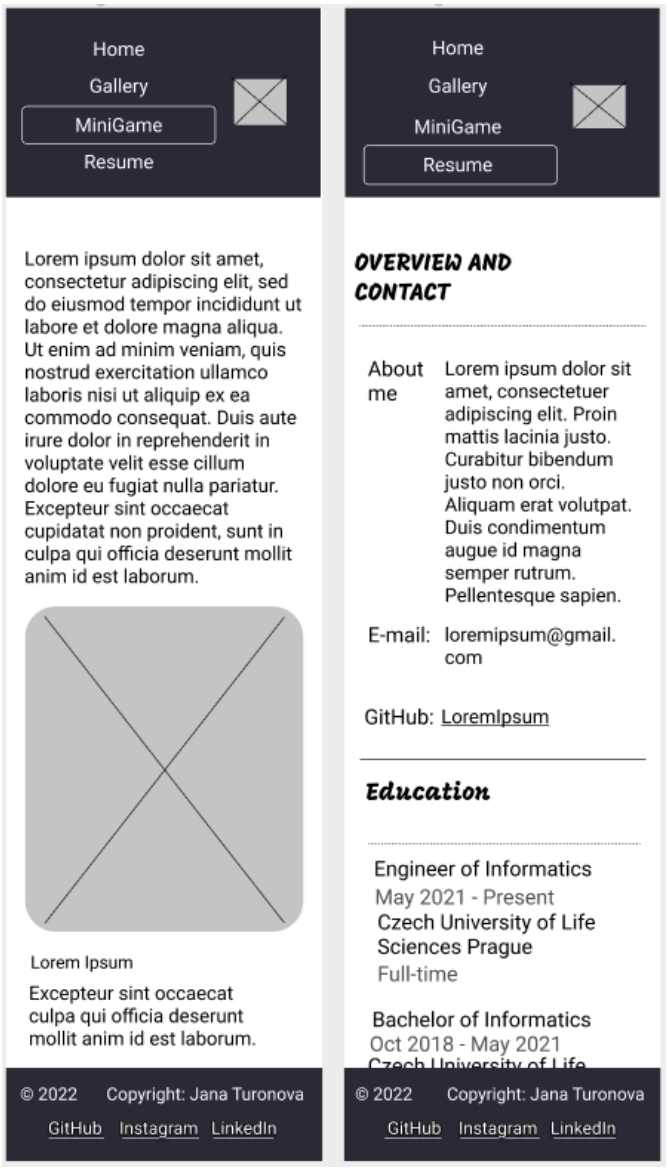


Obrázek 14 - První wireframe – desktopová verze (vlastní zpracování)

Druhý wireframe pak ukazuje galerii obrázků z cest, kde po kliknutí na zmenšený obrázek se zobrazí větší verze tohoto obrázku. Obrázek je možné zavřít pomocí křížku anebo pokud uživatel klikne do prostoru zšedivělé stránky. Tato galerie poskytuje uživatelům možnost prohlédnout si obrázky z cest v plné velikosti. Oba tyto wireframy pomohly navrhnout layout a strukturu těchto sekcí webové stránky a zajistit tak jejich přehlednost a uživatelskou přívětivost.

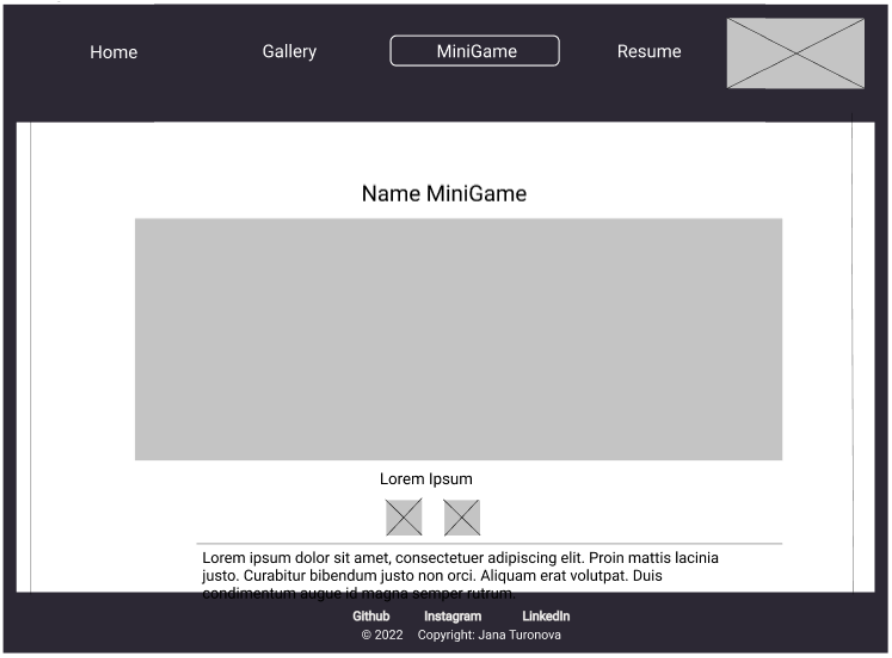


Obrázek 15 - Druhý wireframe – desktopová verze (vlastní zpracování)



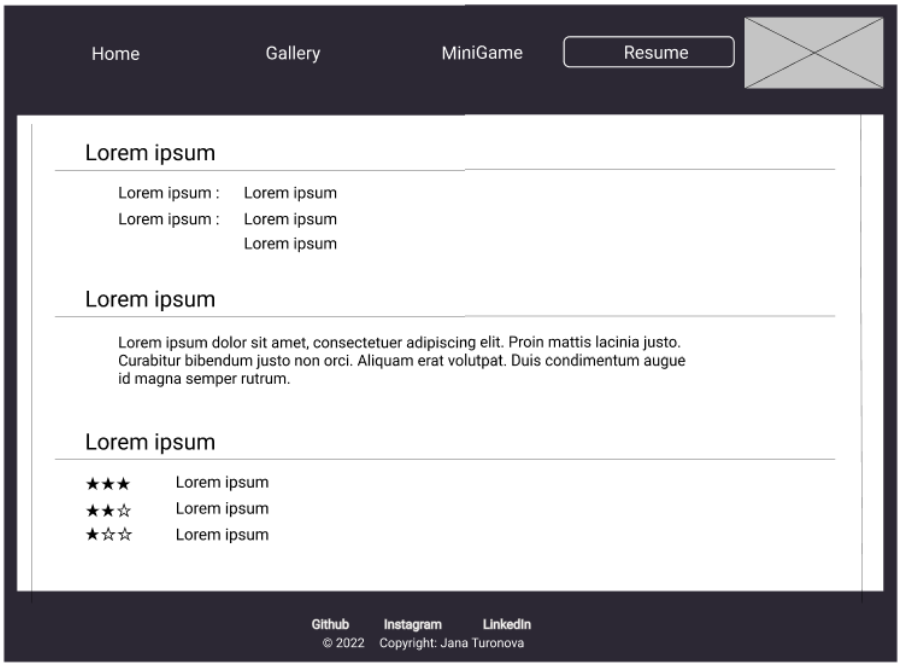
Obrázek 16 - Návrh webové stránky pro mobilní zařízení (vlastní zpracování)

Ve třetím wireframu obsahuje návrh stránky minihry, na které jsou zobrazeny přesné instrukce k dané hře a hra samotná, která je naprogramovaná na jiné webové stránce a vložena do stránky. Důvodem je shodnost verzí u všech tří frameworků.



Obrázek 17 – Třetí wireframe – desktopová verze (vlastní zpracování)

Poslední wireframe pak představuje životopis vývojáře, který umožňuje snadnější vstup na pracovní trh.



Obrázek 18 - Čtvrtý wireframe – desktopová verze (vlastní zpracování)

Pro sjednocení stylu na všech webových stránkách byl použit framework Bootstrap a primárně využita technologie flexbox, která umožňuje snadnou úpravu layoutu stránky při změně velikosti obrazovky.

4.1.3 Hlavička (header)

Hlavička stránky obsahuje navigaci a logo. Pro navigaci v menu je vymezen prostor pomocí elementu "col", který umožňuje prvkům menu libovolně měnit velikost. Pro označení aktuální položky menu je použito ohraničení `border`, jak je vidět na návrzích wireframu. Uživatelé se tak mohou snadno orientovat v menu a rychle se dostat k požadovanému obsahu.

4.1.4 Tělo (body)

Tělo stránky je tvořeno routerem, což je prvek, který umožňuje načítání nových dat a aktualizaci aktuální stránky bez nutnosti jejího opětovného načtení. Toto je důležité pro Single Page Application (SPA).

Tělo stránky také obsahuje element `main`, který mění barvu textu na černou a je mu přidělena minimální výška pomocí výpočtu `calc(80% - 100px)`, což znamená, že její výška je 80% výšky zařízení minus 100 pixelů. Tento výpočet slouží k odečtení výšky patičky z výšky `main` elementu. Dále je v těle stránky uveden oddíl, který má bílé pozadí, důvodem je tmavá barva pozadí na celé webové stránce.

4.1.5 Patička (footer)

Patička stránky je navržena tak, aby se zobrazovala na spodní části obrazovky po celou dobu, kdy je stránka otevřena. Obsahuje typické informace, jako je aktuální rok a jméno vývojáře. Mimo těchto informací sekce patičky obsahuje připnuté odkazy na osobní účty vývojáře, jako je GitHub, Instagram a LinkedIn. Odkazy umožňují uživatelům snadno navázat kontakt s vývojářem nebo se dozvědět více o jeho práci.

4.1.6 Grafický vzhled

Stránky jsou navrženy s jednotným grafickým vzhledem, který se skládá z bílé, indigo a tmavě šedé barvy. Pro zvýraznění funkcí je použita barva námořnická modř, zatímco pro označení aktuální polohy uživatele v menu se používá bílá barva rámečku o šířce 2px a zaoblením 6px.

4.2 Struktura projektu

4.2.1 Client-side

Aplikace na straně klienta byly implementovány jako Single Page Application (SPA) a byly vyvinuty pomocí JavaScriptových frameworků Vue, React a Angular. Všechny tyto frameworky využívají TypeScript, což je nadstavba jazyka JavaScript, který poskytuje zvýšenou možnost statického typování a další vlastnosti, které zlepšují čitelnost a srozumitelnost kódu.

4.2.2 Server-side

Server se nachází na webové stránce https://diplomkaserver.netlify.app/. Na serveru se věnuje zvláštní pozornost kódu JSON, který slouží k specifikaci nadpisu, popisu a URL adresy pro jednotlivé obrázky. Tyto obrázky jsou generovány do jednoho konkrétního souboru na serveru, z něhož mohou ostatní webové stránky čerpat potřebné informace a tím získat přístup ke všem obrázkům. Na serveru se taktéž nachází text k webovým stránkám, který je napsán v Markdownu, soubory končí příponou .md.

Pro přístup dalších stránek k souborům na serveru bylo nutné povolit politiky CORS, protože bude server zveřejněný na Netlify, je možné použít netlify.toml, kde je nutné definovat hlavičku a následně proměnnou, v našem případě se jedná o Access-Control-Allow-Origin, který povolí politiky CORS.

```
[[headers]]
  for = "/*"
  [headers.values]
    Access-Control-Allow-Origin = "*"
```

4.2.3 Propojení Server-side a Client-side

Server a klient jsou navrženy tak, aby bylo možné snadno implementovat na více různých aplikacích stejné části kódu, aniž by bylo nutné je duplikovat.

Pro práci s client side a server side byla vytvořena specifická komponenta pro jednotlivé JavaScriptové frameworky. Tyto komponenty získávají informace z API na serveru z JSON souborů a následně implementují potřebné údaje, jako je název obrázku a samotný obrázek pomocí URL, na webové stránky.

4.2.4 API server

Aplikační programové rozhraní (API) server je implementován jako statické textové soubory formátu JSON.

Tato implementace představuje repositář, který slouží k nastavení a správě dat v blokové struktuře bez přítomnosti aktivní logiky. Tento repositář je důležitým prvkem pro poskytování a uchovávání informací pro klientské aplikace, které se připojují k API serveru, a umožňuje jim rychlý, efektivní a bezproblémový přístup k datům.

4.2.5 Spuštění aplikace

Webové projekty byly uloženy v repositářích GitHub, odkud byly automaticky nasazeny na aplikaci Netlify, která umožňuje integraci s platformami pro vývoj webových aplikací jako je GitHub, GitLab a Bitbucket.

Výhodou Aplikace Netlify je, že při každém pushnutí do repositáře se automaticky nasadí i webový projekt, což šetří čas. Pokud se stane, že webový projekt se špatně nasadí, spustí se poslední úspěšná verze projektu.

Hlavní nevýhodou aplikace Netlify je omezení při spravování a konfiguraci serveru. Aplikace Netlify poskytuje pouze základní možnosti nastavení a úprav serveru. Pro jednoduší projekty jsou tyto nastavení dostačující, avšak pro složitější projekty mohou být limitující.

4.3 Testování JS frameworků

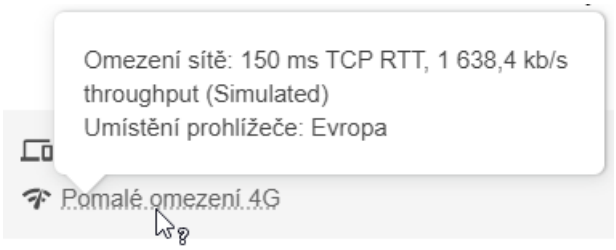
Pro testování stránek byl využit online nástroj PageSpeed Insights, který lze nalézt na webové adrese https://pagespeed.web.dev/. Jedná se o velmi moderní nástroj, který testuje webové stránky z pohledu uživatelů a poskytuje návrhy, jak zlepšit běh stránky. Lze zde najít informace o SEO a o přístupnosti webové stránky. Pro všechny frameworky ale platí, že by měly mít stejné skóre přístupnosti a skóre SEO, proto již nadále s touto informací není nikde pracováno.

Nástroj PageSpeed Insights poskytuje celkové výkonnostní skóre webové stránky. Není možné hodnotit webové stránky jen podle celkového výkonnostního skóre, pro testování se pracovalo se všemi metriky, který web poskytne.

Pro simulaci načtení stránek bylo použito zařízení Moto G4 pro mobilní zařízení, které poskytuje webová stránka PageSpeed Insights a pro desktopová zařízení bul použit notebook značky Dell označení G15.

Při simulaci byly naměřené rozdílné hodnoty metrik, důvodem jsou síťové podmínky při zátěži síťového připojení.

Pro mobilní zařízení byly naměřeny hodnoty s větším rozptylem, důvodem je pak používání virtuálního zařízení ze stránky PageSpeed Insights. Dále bylo důležité zkontrolovat, kde daný test probíhal – PageSpeed může běžet v Severní Americe, Evropě nebo Asii, proto byly brány v potaz pouze výsledky z Evropy.



Obrázek 19 - Podrobnosti běhu testu PageSpeed (vlastní zpracování)

Vzhledem k možným rozdílným hodnotám, z důvodu síťového připojení, bylo měření vždy pro každou webovou stránku opakováno pětkrát, aby se eliminovaly náhodné vlivy v měření. Zapisování výsledků probíhalo v programu Microsoft Excel, kde byly provedeny funkce MIN, MAX a AVERAGE. Získané hodnoty z jednotlivých webových stránek byly vyhodnoceny a vzájemně porovnány.

4.3.1 Použité metriky

Nejprve je důležité vysvětlit, co jednotlivé metriky znamenají a jaký mají význam při testování webových stránek. Jaký vliv mají metriky na uživatele a jak různé faktory mohou ovlivnit jejich hodnoty.

First Contentful Paint (FCP)

FCP, v překladu první vykreslení obsahu, měří čas od okamžiku, kdy se stránka začne načítat, do okamžiku, kdy je na obrazovce vykreslena jakákoli část obsahu stránky. (51)

Obrázek 20 - Výstup z PageSpeed (vlastní zpracování)

Obrázek výše znázorňuje postupné načítání webové stránky. Ve webové aplikaci se jedná o druhý snímek, kdy se načte barva pozadí.

Time to Interactive (TTI)

TTI měří dobu, než se webová stránka stane plně interaktivní a uživatel může začít používat její funkce. TTI je velmi ovlivnitelné interakcí uživatelů, tzn. pokud uživatel na stránce klikne nebo provede jinou akci, zatímco se stále vypočítává hodnota, může to ovlivnit naměřenou hodnotu TTI. Hodnota TTI je udávána v sekundách. (52)

Webové stránky zatím nemají skoro žádnou návštěvnost, proto lze využít metriku TTI, která bude naměřená online nástrojem PageSpeed.

Speed Index

Metrika Speed Index ukazuje, jak rychle se viditelně vyplní obsah stránky. Nezahrnuje pouze dobu načítání prvního obsahu, ale celou dobu načítání stránky včetně dynamických komponent. Hodnota Speed Indexu je vyjádřena v sekundách a reprezentuje dobu, kterou trvá, než se stránka zobrazí uživateli.

Total Blocking Time (TBT)

Metrika měří dobu, po kterou je webová stránka blokována hlavním vláknem, jednotkou této metriky jsou milisekundy. Během této doby se na stránce nedá dělat nic jiného a uživatel musí čekat, až se stránka uvolní. (53) Nižší hodnota znamená, že stránka bude méně blokující a bude reagovat rychleji na uživatelské akce.

Largest Contentful Paint (LCP)

LCP, metrika říká, jak dlouho trvá načíst největší obsah vzhledem k tomu, kdy se poprvé začala stránka načítat. (54) Stejně jako u FCP je metrika udávána v sekundách. Vzhledem k tomu, že je na stránkách používaný Markdown, který se nejprve musí načíst ze

serveru, bude se jednat o největší obsah. Na stránce s galerií je nutno předpokládat, že obrázky se budou načítat nejdéle, vzhledem k velkému objemu dat.

Cumulative Layout Shift (CLS)

Kumulativní změna rozvržení měří stabilitu layoutu stránky během načítání. Hodnota je znázorněna jako skóre posunu rozložení. Vyšší hodnota znamená, že prvky na stránce se během načítání mohou pohybovat, a to může být rušivé pro uživatele. Nižší hodnota znamená stabilnější a předvídatelnější uživatelskou zkušenost. (55)

Metrika CLS bude mít vyšší hodnotu ve všech třech frameworcích na stránce Gallery, vzhledem k objemným obrázkům, které se načítají zdlouhavě.

4.3.2 React

Testování nejprve proběhlo pro webovou aplikaci vytvořenou s frameworkem React, celková měření pro mobilní i desktopovou verzi lze dohledat v příloze.

Mobilní verze

Home

Velmi dobré hodnoty byly zaznamenány pro metriky FCP a Speed Index, což znamená, že stránky se velmi rychle načítají a uživatel může interagovat s obsahem téměř okamžitě. Metrika LCP je mírně pomalejší, ale stále se drží na dobré úrovni. Horší hodnota LCP 3.98 s je kvůli načítání Markdownu ze serveru. TTI a CLS jsou na dobré úrovni, což znamená, že stránky poskytují uživatelsky přívětivé prostředí. Maximální hodnoty jsou většinou velmi podobné průměrným hodnotám metrik. Minimální hodnoty jsou také velmi blízké průměrným hodnotám, což je další pozitivní znak.

Gallery

Ukazuje větší rozdíl mezi minimálními a maximálními hodnotami, důvodem je jedno měření, při kterém mohlo dojít k síťovým problémům. Nejlepší výsledky jsou opět pro metriky FCP a Speed Index. Metrika LCP se zde již drží na nižší úrovni a TTI a CLS jsou mírně horší. Maximální hodnota metriky TBT je vysoká, důvodem je velké množství obrázků, které se snaží načíst

MiniGame

Nejlepší výsledky jsou zde pro metriky LCP a CLS, což je velmi pozitivní, protože tyto metriky se týkají uživatelské zkušenosti. Metrika FCP a Speed Index jsou také na dobré úrovni, ale maximální hodnota metriky TBT je vysoká. Důvodem může být načítání hry z jiné webové stránky. TBT může mít vliv při používání hry a odradit uživatele od dalšího hraní.

Resume

Nejlepší hodnoty jsou zaznamenány pro metriky FCP a Speed Index, což znamená, že uživatel může interagovat s obsahem téměř okamžitě. Metrika LCP je mírně pomalejší, ale stále na dobré úrovni. Hodnoty pro metriky TTI a CLS jsou velmi dobré a maximální a minimální hodnoty jsou blízko u sebe. Maximální hodnota TBT na stránce Resume je 270 milisekund, což značí zpoždění při interakci s některými prvky.

Home		Gallery		MiniGame		Resume	
MAX							
FCP	1.9	FCP	1.9	FCP	1.9	FCP	1.9
Speed Index	2.8	Speed Index	2.7	Speed Index	2.7	Speed Index	2.3
LCP	4.4	LCP	6.1	LCP	3.2	LCP	3.1
TTI	2.3	TTI	2.4	TTI	3.0	TTI	2.5
TBT	160	TBT	170	TBT	190	TBT	270
CLS	0.511	CLS	0.265	CLS	0.042	CLS	0.108
MIN							
FCP	1.9	FCP	1.9	FCP	1.9	FCP	1.9
Speed Index	2.0	Speed Index	1.9	Speed Index	1.9	Speed Index	1.9
LCP	3.7	LCP	3.1	LCP	2.9	LCP	2.9
TTI	2.1	TTI	2.3	TTI	2.3	TTI	2.3
TBT	50	TBT	80	TBT	70	TBT	60
CLS	0.310	CLS	0.081	CLS	0.038	CLS	0.108
AVERAGE							
FCP	1.9	FCP	1.9	FCP	1.9	FCP	1.9
Speed Index	2.34	Speed Index	2.28	Speed Index	2.26	Speed Index	2.06
LCP	3.98	LCP	3.74	LCP	2.96	LCP	2.94
TTI	2.26	TTI	2.36	TTI	2.5	TTI	2.4
TBT	90	TBT	132	TBT	148	TBT	148
CLS	0.378	CLS	0.1572	CLS	0.0388	CLS	0.108

Tabulka 2 - Metriky pro React pro mobilní verzi (vlastní zpracování)

Desktopová verze

Stránky mají v průměru dobré výkonnostní charakteristiky a nízké množství kumulativního posunu obsahu (CLS). Výjimku tvoří stránka Gallery, kde se načítá velké množství objemných obrázků a průměrné CLS je 0.19.

Průměrnou odezvu na první načtení stránky (FCP) mají stránky okolo hodnoty 0.5 s, nejrychlejší načtení obsahu (LCP) má stránka MiniGame společně s Resume, hodnota je 0.7 s. Nejvyšší rychlost stránky v průměru (Speed Index) má stránka MiniGame s hodnotou 0.72 s.

TBT hodnoty pro všechny stránky jsou 0 ms, což je velmi dobré. Čím nižší hodnota TBT, tím rychlejší a plynulejší bude pro uživatele načtení stránky. Průměrná hodnota TTI je kolem 0.5 s a MiniGame stránka má nejnižší hodnotu TTI, což znamená že uživatel může interagovat se stránkou skoro okamžitě.

Celkově jsou výsledky uspokojivé a návštěvníci by neměli mít problémy s načítáním a používáním stránek.

Home		Gallery		MiniGame		Resume	
MAX							
FCP	0.6	FCP	0.6	FCP	0.6	FCP	0.5
Speed Index	1.0	Speed Index	0.9	Speed Index	0.8	Speed Index	1.1
LCP	1.0	LCP	2.4	LCP	0.7	LCP	0.7
TTI	1.0	TTI	0.5	TTI	0.7	TTI	0.5
TBT	0	TBT	0	TBT	0	TBT	0
CLS	0.008	CLS	0.229	CLS	0.008	CLS	0.043
MIN							
FCP	0.5	FCP	0.5	FCP	0.5	FCP	0.5
Speed Index	0.6	Speed Index	0.7	Speed Index	0.5	Speed Index	0.5
LCP	1.0	LCP	1.2	LCP	0.7	LCP	0.7
TTI	0.5	TTI	0.5	TTI	0.5	TTI	0.5
TBT	0	TBT	0	TBT	0	TBT	0
CLS	0.004	CLS	0.144	CLS	0.007	CLS	0.043
AVERAGE							
FCP	0.52	FCP	0.52	FCP	0.5	FCP	0.5
Speed Index	0.74	Speed Index	0.82	Speed Index	0.72	Speed Index	0.82
LCP	1	LCP	1.76	LCP	0.70	LCP	0.7
TTI	0.62	TTI	0.5	TTI	0.60	TTI	0.5
TBT	0	TBT	0	TBT	0	TBT	0
CLS	0.0058	CLS	0.191	CLS	0.0072	CLS	0.043

Tabulka 3 - Metriky pro React pro desktopovou verzi (vlastní zpracování)

4.3.3 Angular

Mobilní verze

Home

Stránka Home dosahuje průměrných výsledků metrik, přičemž nejhorší hodnotu má v metrice LCP 4.28 s, která měří čas prvního načtení hlavního obsahu stránky, a TBT 448 ms, která měří dobu trvání interaktivního načtení stránky. V metrice CLS, která měří stabilitu rozložení obsahu, má stránka hodnotu v průměru 0.5616, což může mít za následek špatný uživatelský zážitek, důvodem je Markdown, který se načítá ze serveru.

Gallery

Stránka Gallery má vysokou rychlost načítání, středně vysoký TBT s hodnotou 296 ms a nízký CLS s hodnotou 0.145, což umožňuje hladkou interakci s uživatelem. LCP je rychlejší než na stránce Home.

MiniGame

Stránka MiniGame má podobné metriky jako stránka Gallery s výjimkou LCP, TTI a TBT, které jsou mírně pomalejší. Tato stránka se ale primárně zaměřuje na hru, takže vysoká hodnota TBT může být tolerována.

Resume

Stránka Resume má pomalejší načítání než ostatní stránky a vysoký TBT s hodnotou 474 ms. CLS je také vyšší než u ostatních stránek kromě stránky Home. Nicméně, Speed Index je rychlý a umožňuje uživateli rychlou interakci se stránkou.

Home		Gallery		MiniGame		Resume	
MAX							
FCP	2.1	FCP	2.1	FCP	2.1	FCP	2.1
Speed Index	3.0	Speed Index	2.7	Speed Index	2.7	Speed Index	2.9
LCP	4.6	LCP	3.9	LCP	3.9	LCP	4.1
TTI	3.2	TTI	3.2	TTI	3.5	TTI	3.3
TBT	500	TBT	440	TBT	450	TBT	580
CLS	0.772	CLS	0.277	CLS	0.326	CLS	0.426
MIN							
FCP	2.0	FCP	1.9	FCP	2.1	FCP	2.0
Speed Index	2.3	Speed Index	2.3	Speed Index	2.1	Speed Index	2.2
LCP	3.8	LCP	3.2	LCP	3.3	LCP	3.4
TTI	2.7	TTI	2.4	TTI	2.9	TTI	2.9
TBT	340	TBT	150	TBT	260	TBT	350
CLS	0.481	CLS	0.030	CLS	0.038	CLS	0.138

AVERAGE							
FCP	2.1	FCP	2	FCP	2.1	FCP	2.02
Speed Index	2.66	Speed Index	2.46	Speed Index	2.4	Speed Index	2.5
LCP	4.28	LCP	3.56	LCP	3.62	LCP	3.76
TTI	2.98	TTI	2.9	TTI	3.2	TTI	3.08
TBT	448	TBT	296	TBT	356	TBT	474
CLS	0.5616	CLS	0.1454	CLS	0.1538	CLS	0.253

Tabulka 4 - Metriky pro Angular pro mobilní verzi (vlastní zpracování)

Desktopová verze

Lze pozorovat, že stránky Home a MiniGame mají všechny měřené hodnoty v oblasti rychlosti na úrovni 0.5 až 0.8 sekundy, což je velmi dobré. Zatímco Gallery a Resume jsou pomalejší s měřenými hodnotami až 3.6 sekundy.

Kromě rychlosti je také brán zřetel na metriky jako LCP, TTI a TBT, které jsou také velmi důležité pro uživatelskou zkušenost. Home a MiniGame mají nejlepší výsledky v těchto oblastech, což znamená, že uživatelé těchto stránek se mohou rychleji dostat k požadovaným informacím. Gallery a Resume mají horší výsledky v těchto oblastech, což může znamenat, že uživatelé budou muset delší dobu čekat na zobrazení obsahu.

Je-li podrobně zkoumána metrika CLS (vizuální stabilita), je vidět, že stránky Home a Gallery mají nejhorší výsledky, což znamená, že se obsah na stránce může přesouvat a uživatele to může rušit. MiniGame a Resume mají nejlepší výsledky v této oblasti, takže uživatelé nebudou rušeni pohybujícím se obsahem.

Home		Gallery		MiniGame		Resume	
MAX							
FCP	0.5	FCP	0.5	FCP	0.8	FCP	2.0
Speed Index	1.0	Speed Index	1.0	Speed Index	1.9	Speed Index	2.2
LCP	1.1	LCP	2.9	LCP	1.0	LCP	3.6
TTI	0.7	TTI	0.7	TTI	0.8	TTI	2.7
TBT	100	TBT	40	TBT	120	TBT	210
CLS	0.127	CLS	0.356	CLS	0.008	CLS	0.138
MIN							
FCP	0.5	FCP	0.5	FCP	0.5	FCP	0.5
Speed Index	0.5	Speed Index	0.6	Speed Index	0.5	Speed Index	0.7
LCP	1.1	LCP	1.3	LCP	0.9	LCP	0.9
TTI	0.6	TTI	0.6	TTI	0.7	TTI	0.7
TBT	20	TBT	30	TBT	0	TBT	30
CLS	0.004	CLS	0.086	CLS	0.006	CLS	0.063

AVERAGE							
FCP	0.5	FCP	0.5	FCP	0.6	FCP	0.8
Speed Index	0.74	Speed Index	0.76	Speed Index	0.94	Speed Index	1.04
LCP	1.1	LCP	1.9	LCP	0.92	LCP	1.54
TTI	0.66	TTI	0.66	TTI	0.76	TTI	1.18
TBT	42	TBT	34	TBT	50	TBT	76
CLS	0.0532	CLS	0.1658	CLS	0.0064	CLS	0.085

Tabulka 5 - Metriky pro Angular pro desktopovou verzi (vlastní zpracování)

4.3.4 Vue

Mobilní verze

Home

První obsah na stránce Home se zobrazí po 1.7 sekundách od načtení stránky (FCP). Celkový čas načítání stránky (Speed Index) je poměrně dobrý, 2.06 sekundy, ale největší obsah na stránce (LCP) se načte poměrně pomalu, po 3.72 sekundách. Stránka se stane interaktivní (TTI) velmi rychle, ale celkový čas blokování (TBT) je poměrně vysoký s hodnotou 104 ms, což může vést ke špatnému uživatelskému zážitku. Tento efekt může být dále zhoršen poměrně vysokým celkovým posunem rozvržení (CLS), důvodem této velké hodnoty je Markdown.

Gallery

Obsah na stránce se zobrazí stejně rychle jako na stránce Home. Celkový čas načítání stránky (Speed Index) je mírně horší než na stránce Home s hodnotou 2.32 sekundy. Největší obsah na stránce (LCP) se načítá pomalu, po 3.04 sekundách, důvodem jsou objemné obrázky. Stránka se stane interaktivní po 2.02 sekundách (TTI), ale celkový čas blokování (TBT) je lepší než na stránce Home s hodnotou 79.4 ms. Celkový posun rozvržení (CLS) je s hodnotou 0.279 lepší než na stránce Home.

MiniGame

Největší obsah na stránce (LCP) se načítá rychleji než na předchozích stránkách, po 2.61 sekundách. Stránka se stane interaktivní po 2.22 sekundách (TTI) a celkový čas blokování (TBT) je také lepší než na stránkách Home a Gallery, s hodnotou 74.8 ms. Celkový posun rozvržení (CLS) je nejlepší ze všech stránek, s hodnotou 0.141, důvodem může být nejmenší obsah textu, který se má vygenerovat.

Resume

První obsah na stránce Resume se také zobrazí po 1.7 sekundách od načtení stránky (FCP), stejně jako na předchozích stránkách. Celkový čas načítání stránky (Speed Index) je opět podobný s hodnotou 2.34 sekundy. Největší obsah na stránce (LCP) se načítá poměrně rychle, po 2.5 sekundách. Stránka se stane interaktivní po 1.9 sekundách (TTI), ale celkový čas blokování (TBT) je stejně jako na stránce Home poměrně vysoký s hodnotou 104 ms. Celkový posun rozvržení (CLS) je s hodnotou 0.19 mírně horší než na stránce MiniGame.

Home		Gallery		MiniGame		Resume	
MAX							
FCP	1.7	FCP	1.7	FCP	1.7	FCP	1.7
Speed Index	2.5	Speed Index	3.1	Speed Index	3.0	Speed Index	2.6
LCP	4.0	LCP	4.3	LCP	2.8	LCP	2.5
TTI	2.1	TTI	2.1	TTI	2.3	TTI	2.1
TBT	120	TBT	90	TBT	100	TBT	160
CLS	0.489	CLS	0.460	CLS	0.141	CLS	0.190
MIN							
FCP	1.7	FCP	1.7	FCP	1.7	FCP	1.7
Speed Index	1.7	Speed Index	1.7	Speed Index	1.7	Speed Index	1.7
LCP	3.2	LCP	2.6	LCP	2.5	LCP	2.5
TTI	1.9	TTI	2.0	TTI	2.1	TTI	1.8
TBT	80	TBT	70	TBT	51	TBT	70
CLS	0.218	CLS	0.161	CLS	0.141	CLS	0.190
AVERAGE							
FCP	1.7	FCP	1.7	FCP	1.7	FCP	1.7
Speed Index	2.06	Speed Index	2.32	Speed Index	2.34	Speed Index	2.34
LCP	3.72	LCP	3.04	LCP	2.61	LCP	2.5
TTI	1.94	TTI	2.02	TTI	2.22	TTI	1.9
TBT	104	TBT	79.4	TBT	74.8	TBT	104
CLS	0.3264	CLS	0.279	CLS	0.141	CLS	0.19

Tabulka 6 - Metriky pro Vue pro mobilní verzi(vlastní zpracování)

Desktopová verze

Průměrná rychlost FCP a Speed Index jsou podobné u všech stránek, což znamená, že uživatelé uvidí obsah stránky poměrně rychle. Avšak, hodnoty LCP (Largest Contentful Paint) jsou výrazně vysoké na stránkách Gallery a Home, což může vést ke zpomalení načítání stránky.

Hodnoty TTI (Time to Interactive) jsou poměrně nízké na všech stránkách, což znamená, že uživatelé mohou začít s interakcí se stránkou poměrně rychle. Hodnoty TBT (Total Blocking Time) jsou nízké na všech stránkách kromě stránky Resume, kde jsou hodnoty výrazně vyšší. Tato hodnota může zapříčinit zpoždění při načítání stránky.

Hodnota CLS (Cumulative Layout Shift) na všech stránkách je relativně nízká, z toho vyplývá, že uživatelé nebudou překvapeni neočekávanými posuny prvků na stránce během načítání.

Celkově lze říci, že webové stránky mají poměrně rychlé načítání s nízkými hodnotami CLS.

Home		Gallery		MiniGame		Resume	
MAX							
FCP	0.4	FCP	0.4	FCP	0.4	FCP	0.4
Speed Index	0.8	Speed Index	1.0	Speed Index	1.0	Speed Index	1.0
LCP	1.0	LCP	2.2	LCP	0.7	LCP	0.7
TTI	0.5	TTI	0.5	TTI	0.5	TTI	0.5
TBT	30	TBT	10	TBT	10	TBT	40
CLS	0.003	CLS	0.160	CLS	0.061	CLS	0.087
MIN							
FCP	0.4	FCP	0.4	FCP	0.4	FCP	0.4
Speed Index	0.4	Speed Index	0.7	Speed Index	0.7	Speed Index	0.6
LCP	0.9	LCP	1.0	LCP	0.7	LCP	0.6
TTI	0.4	TTI	0.4	TTI	0.4	TTI	0.4
TBT	0	TBT	0	TBT	0	TBT	0
CLS	0.003	CLS	0.106	CLS	0.060	CLS	0.087
AVERAGE							
FCP	0.4	FCP	0.4	FCP	0.4	FCP	0.4
Speed Index	0.64	Speed Index	0.9	Speed Index	0.84	Speed Index	0.78
LCP	0.92	LCP	1.54	LCP	0.70	LCP	0.68
TTI	0.46	TTI	0.44	TTI	0.42	TTI	0.46
TBT	10	TBT	2	TBT	2	TBT	16
CLS	0.003	CLS	0.137	CLS	0.0602	CLS	0.087

Tabulka 7 - Metriky pro Vue pro desktopovou verzi(vlastní zpracování)

4.4 Porovnání rozdílu mezi jednotlivými frameworky

4.4.1 Mobilní verze

Na základě naměřených metrik lze vypozorovat, že framework Vue konzistentně načítá obsah rychleji (FCP) o několik desetin sekundy, následovaný React a Angular na posledním místě.

Nejrychlejší Speed Index (SI), což je doba, kdy se zobrazí obsah stránky na celou šířku, tak framework React vykazuje nejlepší výsledky, s výjimkou stránky Home, kdy nejrychlejší načítání má Vue. Nejpomalejším Frameworkem v SI je Angular.

V oblasti největšího prvku na stránce (LCP) byl Vue nejrychlejší, následovaný Reactem. V případě stránky Gallery však Angular s hodnotou 3.56 s dosáhl nejlepšího výsledku. Když přijde na čas, kdy je stránka plně interaktivní (TTI), opět byl Vue nejrychlejší, s pouze o málo pomalejším Reactem. Angular byl nejpomalejší v této kategorii.

Total Blocking Time (TBT), což je doba, kdy je stránka blokována během načítání, tak Angular ukazuje nejhorší výsledky ve všech testovaných stránkách. Nejlepší výsledky v této kategorii dosahuje Vue.

Při pohledu na Cumulative Layout Shift (CLS), který značí posun stránek během načítání. Vykazuje znovu nejlepší výsledky framework Vue. Na druhém místě je React a na posledním místě Angular.

4.4.2 Desktopová verze

Porovnání frameworků pro desktopovou verzi prohlížeči lze vypozorovat, že nejlepších výsledků pro stránku Home dosahuje Framework Vue pro všechny metriky, kromě metriky TBT. Pro metriku TBT dosahuje nejlepšího výsledku framework React, který dosahuje jako druhý nejlepších výsledků, ačkoliv první načtení obsahu trvá v průměru o něco déle něž Angularu, který dosáhl nejhorších metrik. Největší rozdíly lze vypozorovat u posunu stránek během načítání (CLS), který u Angularu je 0.0532, u Vue 0.003 a u Reactu 0.0058, a u Speed Indexu (SI), kdy u Angularu je v průměru 1.1s, u Vue 0.64 s a u Reactu 0.74 s.

U stránky Gallery má opět nejlepší výsledky framework Vue ve všech výkonnostních metrikách, kromě Speed Indexu, kde je jeho hodnota 0.9 s a zaostává za Reactem 0.82 s i Angularem 0.76 s. Framework React má lepší výsledky u stránky Gallery než Angular, největší rozdíl v metrikách je možné si povšimnout u TBT, doba, kdy je stránka blokována

během načítání – React má hodnotu 0 ms a Angular hodnotu 34 ms. Angular je v metrikách LCP, TTI a TBT, velmi pomalý a jeho výsledky jsou vždy o několik desetin horší.

Pro stránku MiniGame lze pozorovat, že nejlepších výsledků dosahuje framework Vue u metrik FCP, LCP a TTI, ačkoliv u metriky CLS dosahuje nejhoršího výsledku ze všech frameworků. Framework React dosahuje nejlepšího výsledku u metriky TBT, kde je hodnota 0 ms a u metriky Speed index, kde je hodnota 0.72 s, zatímco v ostatních metrikách zaostává za frameworkem Vue. Framework Angular dosahuje nejlepšího výsledku u metriky CLS, avšak jinak dosahuje nejhorších výsledků ze všech frameworků, největší rozdíl jdou zpozorovat u metriky TBT, kde dosahuje hodnoty 50 ms, což je výrazně horší než v případě Reactu a Vue.

Nejlepší výsledky z hlediska výkonu pro stránku Resume dosahuje framework Vue pro metriky FCP, Speed Index, LCP a TTI, s výjimkou metriky TBT a CLS, kde React dosahuje lepšího výsledku. Framework React dosahuje výsledků ostatních metrik někde mezi Vue a Angularem. Angular dosahuje opět nejhorších výsledků.

4.5 Analýza porovnání vybraných frameworků

V rámci analýzy a porovnávání moderních frameworků pro vývoj webových aplikací, jako jsou Vue, Angular a React, je nutné brát v úvahu řadu faktorů, včetně jejich podpory prohlížečů a velikosti komunity.

4.5.1 Podpora prohlížečů a velikost komunity

Podpora prohlížečů je klíčovým faktorem při výběru vhodného frameworku pro vývoj webových aplikací. Většina moderních webových aplikací musí být kompatibilní s různými prohlížeči, včetně nejnovějších verzí populárních prohlížečů, jako je Google Chrome, Mozilla Firefox, Safari a další. Proto je důležité, aby vybraný framework nabízel dostatečnou podporu prohlížečů, aby bylo možné vytvářet aplikace, které jsou plně funkční a kompatibilní s různými prohlížeči.

Velikost komunity je dalším klíčovým faktorem, který by měl být zohledněn při výběru frameworku pro vývoj webových aplikací. Velká komunita frameworku může být velmi užitečná pro vývojáře, protože nabízí mnoho zdrojů a nástrojů, jako jsou dokumentace, tutoriály, fóra a diskusní skupiny, které mohou pomoci v řešení problémů a překonávání překážek v průběhu vývoje aplikací.

Při srovnávání těchto frameworků lze pozorovat, že Angular podporuje velmi širokou podporou prohlížečů a využívá jej široká komunita vývojářů. Tento fakt je dán především dlouholetou existencí tohoto nástroje. Tato vlastnost je zvláště užitečná pro vývojáře, kteří si přejí minimalizovat rizika spojená s vývojem aplikací.

React má také širokou komunitu a dobrou podporu prohlížečů, i když oficiálně nepodporuje Internet Explorer 11 a 10, což je rozdíl oproti Angularu.

Vue disponuje menší komunitou než Angular a React, ale tato komunita neustále roste, protože Vue je nejmladším frameworkem v této trojici. Na rozdíl od Reactu podporuje Vue starší verze prohlížečů, ale nemá tak rozsáhlou podporu jako Angular.

4.5.2 Křivka učení

Pro všechny JavaScriptové frameworky je nutné a důležité ovládat základní dovednosti pro tvorbu webových stránek, tedy HTML, CSS a JavaScript.

Následně je důležité mít na paměti, že čím jednodušší uspořádání souboru a komponent ve frameworku, tím je jednodušší orientace a použití frameworku. Při pohledu na obrázky

v kapitolách o jednotlivých frameworkách, je možné si všinout, že Vue se příliš neliší od postavení základní aplikace. Naproti tomu Angular rozděluje HTML, CSS a JavaScript do samostatných souborů pro každou komponentu, což může být pro úplné začátečníky velmi matoucí a kód může být nepřehledný.

Důležitým faktorem při učení je také syntaxe. Vue má čistou syntaxi a používá podobný koncept jako v nativním JavaScriptu, což usnadňuje učení a použití pro začátečníky. React používá JSX, což pro mnohé vývojáře může tvořit překážku, nicméně React dovoluje psát také ve funkcionálním stylu.

4.5.3 Měření spotřeby zdrojů

Měření spotřeby zdrojů je jedním z klíčových faktorů pro zajištění rychlého a efektivního běhu webových aplikací. Pro měření spotřeby zdrojů v rámci frameworků React, Vue a Angular byly využity nástroje React Developer Tools pro React, Vue.js devtools pro Vue a Angular DevTools pro Angular.

Při měření spotřeby zdrojů bylo nutné přejít na záložku "Performance" a zde spustit test. Test probíhal tak, že byly „proklikány“ všechny stránky a následně se testování zastavilo a zobrazil se výsledek. Všechny testy trvaly 10 sekund, určitý čas se strávil u každé stránky, konkrétně u Galerie 4 s, MiniGame 2 s, Resume 2 s a Home 2 s.

Zde je tabulka, kde byly jednotlivé frameworky očíslovány podle umístění pro rychlejší přehlednost, dle podpory prohlížečů a velikosti komunity. Ve sloupci spotřeba zdrojů se nachází uvedené hodnoty pro jednotlivé frameworky, hodnoty jsou v MB.

Frameworky	Podpora prohlížečů	Velká komunita	Spotřeba zdrojů
Angular	1	2	7.7–9.8
React	3	1	7.5–8.7
Vue	2	3	7.1–8.6

Tabulka 8 - Porovnání komunity, podpory prohlížečů a spotřeby zdrojů (vlastní zpracování)

5 Výsledky

Na základě provedené komparace frameworků ve vlastní práci a dodatečného zjišťování bylo možné analyzovat potenciální využití těchto vybraných frameworků. Na možnosti využití je možné nahlížet z různých perspektiv.

5.1 Potřeby aplikace

React, Vue a Angular jsou tři z nejpopulárnějších webových frameworků, které umožňují tvorbu vyspělých webových aplikací. Podle uvedených metrik v kapitole 4.3.1 lze doporučit vhodnost použití těchto frameworků na základě požadavků aplikace. Zde je také uvedeno doporučení pro vytvořenou aplikaci.

5.1.1 Doporučení pro vytvořenou aplikaci

Vytvořený testovací prototyp byla menší aplikace s menším počtem komponent, jako je například Markdown komponenta, která slouží k načítání Markdown souboru ze serveru.

Pokud je prioritou webové aplikace rychlost načítání stránky, interaktivita a také minimalizování času, po který je stránka blokována, je určitě nejlepší volbou framework Vue, protože dosahuje nejlepších výsledků v těchto kategoriích na všech testovaných stránkách. Tento framework získal nejlepší výsledky v metrikách FCP, LCP, TTI a CLS na všech testovaných stránkách. Pokud je však důležité, aby byla zobrazována celá stránka co nejrychleji, framework React je nejrychlejší při načítání obsahu na celou šířku (SI) na všech stránkách kromě Home, kde se umístil na druhém místě za Vue.

Pokud se zaměříme na desktopovou verzi prohlížeče, Vue je stále nejlepší volbou pro všechny metriky na stránce Home a MiniGame. Na druhé straně, pokud jsou TBT a SI prioritou, může být nejlepší volbou React. Angular se ukázal jako nejhorší volba v téměř všech metrikách, vyjma metriky LCP na stránce Gallery, kde dosáhl nejlepšího výsledku.

5.1.2 Doporučení dle velikosti aplikace

Jak je možné si povšimnout z výsledků komparace metrik jednotlivých frameworků lze usoudit, že pro implementaci menších aplikací je nejvhodnějším frameworkem Vue. Tento framework se vyznačuje rychlostí a flexibilitou, což jsou klíčové faktory pro menší projekty, jako jsou osobní stránky nebo aplikace pro správu úkolů nebo poznámek.

Pro středně velké aplikace je doporučeno použít framework React. React se zaměřuje na rychlou odezvu UI a zpracování většího množství dat. Jeho modulární architektura a přístup založený na komponentech zajišťuje snadné rozšiřování aplikace a umožňuje vývojovému týmu snadno přizpůsobit aplikaci aktuálním požadavkům. Je vhodný tedy pro stránky jako jsou online prodej vstupenek nebo aplikace na rezervace stolů.

Angular se ukázal ve výsledcích malého projektu jako nejhorší možnou volbou, důvodem je jeho robustnost a komplexnost. Je vhodný pro velké projekty, které vyžadují nejvyšší úroveň výkonu, škálovatelnosti a bezpečnosti. Jako je například Online bankovnictví nebo velká aplikace s mnoha produkty a uživateli.

5.2 Zkušenosti vývojáře

Každý vývojář disponuje individuálními zkušenostmi a dovednostmi v oblasti programování. Při hodnocení rychlosti učení je nutné zohlednit dosavadní zkušenosti, neboť není reálné očekávat, že jedinec, který vládne základním HTML a CSS, dokáže rychle ovládnout JavaScript či nějaký z jeho frameworků.

Proto začínající jedinec, který dosavadně nemá žádné nebo jen střední zkušenosti s JavaScriptem, bude pravděpodobně vyžadovat podstatnou podporu ze strany komunity. Kromě toho bude důležité, aby zvolený framework byl co nejjednodušší, aby se zvýšila pravděpodobnost úspěšného osvojení daných dovedností. Začínající vývojář by mohl zvážit zvolení Reactu na základě několika faktorů. Disponuje velkou komunitou a množstvím knihoven a nástrojů, což může být pro začátečníky velkou výhodou při učení a vývoji. JSX syntaxe, kterou React používá, může být pro začátečníky obtížnější na pochopení než syntaxe Vue. Nicméně, JSX má své výhody, zejména co se týče výkonu a možností úprav. Dále by mohl být React vhodný pro začátečníky, kteří chtějí pracovat na projektech s většími a komplexnějšími aplikacemi. I když React nemusí mít tak vynikající výkonnostní metriky jako Vue.

Pokud začínajícímu vývojáři jde hlavně o nejjednodušší framework na učení a JSX je pro něj příliš složitý a zároveň mu nezáleží na velikosti komunity, tak je určitě tou nejlepší volbou Vue.

Vue lze doporučit všem vývojářům, kteří chtějí nejjednodušší framework na učení a preferují šablonové systémy, kde lze HTML rozšířit o doménově specifický jazyk. Pokud je to potřeba, je možné také využít TypeScript pro zvýšení typové kontroly a dalších výhod.

Nejsložitější framework a nejpomalejší z výsledků výkonnostních metrik je Angular. Někteří vývojáři by mohli preferovat i tento framework, důvodem může být stabilní vývoj bez značné potřeby využívání knihoven třetích stran a striktně typovaný jazyk TypeScript. Tento framework lze doporučit vývojářům, kteří mají zkušenosti v oblasti objektově orientovaných jazyků jako například C#, Java nebo Swift nebo již plně ovládají JavaScript a mají zkušenosti z nějakým z jeho frameworků.

6 Závěr

Tato diplomová práce se zaměřuje na možnosti využití JavaScript frameworků při tvorbě webových single page aplikací. Teoretická část práce vysvětluje základní pojmy související s vývojem webových aplikací jako například programovací paradigma a také základní kameny webu jako je HTML, Markdown, CSS a JavaScript. Dále jsou představeny základní principy SPA a porovnání s tradičními webovými stránkami, tedy MPA. Hlavním tématem teoretické části práce jsou JavaScriptové frameworky, jež představují důležitý nástroj pro vývoj SPA. Jsou zde uvedeny nejpopulárnější frameworky, jako je React, Angular a Vue.js, a jejich možnosti instalace, výhody a nevýhody. V rámci teoretické části práce je možné najít výsledky ankety The State of JS za poslední tři roky. Anketa poskytuje ucelený přehled o trendech a preferencích v oblasti JavaScriptových frameworků a knihoven.

Ve vlastní práci je vytvořen návrh a UI specifikace webové stránky, která je následně realizována ve frameworku Vue, React a Angular. Všechny tři výsledné stránky jsou stejného grafického vzhledu a získávají informace z API na serveru z JSON souborů. Webové projekty byly uloženy v repositářích GitHub, odkud byly automaticky nasazeny na aplikaci Netlify. Následně proběhlo testování pomocí online nástroje PageSpeed Insights, který simuluje test webové stránky z pohledu uživatele. Bylo provedeno testování jak pro desktopovou, tak pro mobilní verzi webových stránek. Každá stránka byla testována pětkrát a pro konečné srovnání výkonu všech frameworků byly použity průměrné hodnoty z testů.

Na základě provedeného testování a dalších kvalitativních kritérií jsou navržena doporučení pro využití frameworků v závislosti na potřebách aplikace a velikosti projektu. Na základě výsledků testů bylo doporučeno použití frameworku Vue pro menší aplikace s důrazem na rychlost načítání a minimalizaci blokování stránky. Framework React se ukázal jako vhodný pro středně velké aplikace s větším počtem komponent a zaměřením na rychlou odezvu UI. Framework Angular byl naopak výsledky testů překonán oběma předchozími frameworky.

Výsledky této práce představují cenné informace pro vývojáře webových aplikací, kteří hledají nejvhodnější framework pro své projekty. Tyto informace mohou významně přispět k optimalizaci výkonu a zlepšení uživatelské zkušenosti webových stránek.

7 Seznam použitých zdrojů

1. Peterka, Jiří. První webová stránka a její historie. *MUZEUM INTERNETU .cz.* [Online] 2011. [Citace: 19. 03 2022.] https://www.muzeuminternetu.cz/webfirst.php3.
2. Introduction of Programming Paradigms. *Geeksforgeeks.* [Online] 17. 11 2022. [Citace: 29. 11 2022.] https://www.geeksforgeeks.org/introduction-of-programming-paradigms/.
3. Functional programming vs. imperative programming. *Microsoft.* [Online] 15. 9 2021. [Citace: 29. 11 2022.] https://learn.microsoft.com/en-us/dotnet/standard/linq/functional-vs-imperative-programming.
4. Differences between Procedural and Object Oriented Programming. *Geeksforgeeks.* [Online] 28. 6 2022. [Citace: 29. 11 2022.] https://www.geeksforgeeks.org/differences-between-procedural-and-object-oriented-programming/.
5. W3C. *HTML A CSS.* [Online] MIT , ERCIM , Keio, Beihang, 2016. [Citace: 15. 03 2022.] https://www.w3.org/standards/webdesign/htmlcss.
6. CASTRO, E. -- HYSLOP, B. *HTML5 a CSS3 : názorný průvodce tvorbou WWW stránek.* Brno : Computer Press, 2012. ISBN 978-80-251-3733-8.
7. W3schools [online]. Norway, 2022 [cit. 2022-03-16]. Dostupné z: https://www.w3schools.com/. [Online]
8. PEHLIVANIAN, Ara a Don NGUYEN. *JavaScript okamžitě.* 2. vydání. Brno : Computer Press, 2021. ISBN 978-80-251-5025-2.
9. Cone, Matt. Markdown Guide. [Online] 2022. [Citace: 14. 05 2022.] https://www.markdownguide.org.
10. Jeffrey, Christopher. MarkedJS: Getting Started. *Marked Documentation.* [Online] (c) 2018. [Citace: 13. 05 2022.] https://marked.js.org.
11. Sass. *SASS: Syntactically Awesome Style Sheets.* [Online] 2006-2022. [Citace: 13. 05 2022.] https://sass-lang.com.
12. Giraudel, Kitty. What's the Difference Between Sass and SCSS? *SitePoint* . [Online] 08. 08 2017. [Citace: 13. 05 2022.] https://www.sitepoint.com/whats-difference-sass-scss/.
13. Bootstrap.com. [Online] 2022. [Citace: 13. 05 2022.] https://getbootstrap.com.
14. GitHub. *GitHub: Where the world builds software.* [Online] 2022. [Citace: 14. 05 2022.] https://github.com.
15. Static vs Dynamic Website. *GeeksforGeeks.* [Online] 19. 08 2022. [Citace: 07. 01 2023.] https://www.geeksforgeeks.org/static-vs-dynamic-website/.

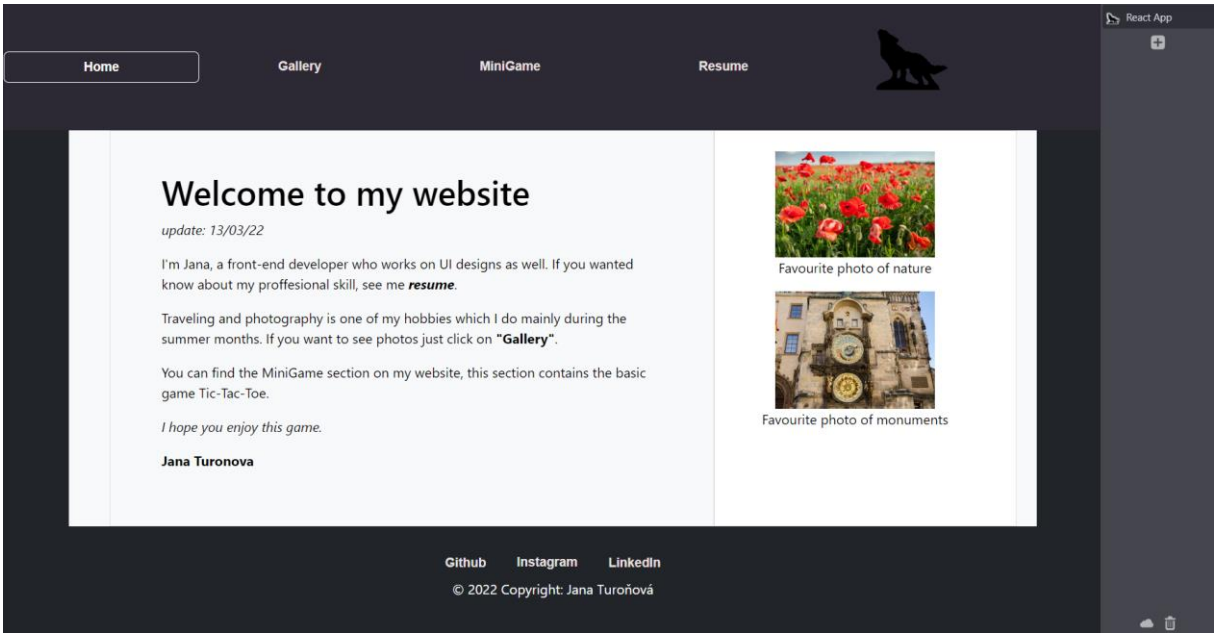
16. Schaefer, Vanessa. What's the Difference Between Dynamic, Responsive, and Adaptive Websites? *Clockwork Design Group.* [Online] 22. 08 2018. [Citace: 07. 01 2023.] https://www.cdgi.com/2018/08/whats-the-difference-between-dynamic-responsive-and-adaptive-websites/.
17. Michálek, Martin. Proč adaptivní, ne responzivní. A proč to responzivní zůstane. *Vzhůru dolů.* [Online] 04. 08 2014. [Citace: 07. 01 2023.] https://www.vzhurudolu.cz/blog/23-adaptivni-responzivni.
18. GRANTHAM, Andrew. What Is SEO / Search Engine Optimization? *Search Engine Land.* [Online] 2022. [Citace: 02. 21 2023.] https://searchengineland.com/guide/what-is-seo.
19. MOZ. The Beginner's Guide to SEO. *Moz.* [Online] 2021. [Citace: 02. 21 2023.] https://moz.com/beginners-guide-to-seo.
20. Halvorson, K., & Rach, M. *Content Strategy for the Web.* CA, USA : New Riders. 2nd edition, 2012. ISBN: 978-0-321-80830-1.
21. Scott, Emmit A. *SPA: Design and Architecture.* NY : Manning Publications Co., 2016. ISBN: 9781617292439.
22. Koďousková, Barbora. Co je to jednostránková webová aplikace (SPA) a kdy ji využít? *RASCONE.* [Online] 24. 07 2021. [Citace: 15. 05 2022.] https://www.rascasone.com/cs/blog/jednostrankova-webova-aplikace-spa.
23. Architektura MVC: definice, struktura, frameworky. *Rascasone.* [Online] 13. 04 2021. [Citace: 18. 05 2022.] https://www.rascasone.com/cs/blog/architektura-mvc-struktura-frameworky.
24. Verma, Anushka. Benefit of using MVC. *GeeksforGeeks.* [Online] 30. 06 2001. [Citace: 18. 05 2022.] https://www.geeksforgeeks.org/benefit-of-using-mvc/.
25. What are MVP and MVC and what is the difference? *Stackoverflow.* [Online] STACK EXCHANGE, 05. 08 2008. [Citace: 18. 05 2022.] https://stackoverflow.com/questions/2056/what-are-mvp-and-mvc-and-what-is-the-difference.
26. Britch, David. Enterprise Application Patterns using Xamarin.Forms. [Online] 2017. [Citace: 18. 05 2022.] https://dotnet.microsoft.com/en-us/download/e-book/xamarin/pdf.
27. JavaScript HTML DOM. *W3Schools.* [Online] Copyright 1999-2023. [Citace: 03. 03 2023.] https://www.w3schools.com/js/js_htmldom.asp.

28. React. *React.* [Online] Meta Platforms, Inc., © 2022. [Citace: 11. 08 2022.] https://reactjs.org.
29. Savkin, Victor. Understanding Angular Ivy: Incremental DOM and Virtual DOM. *Medium.* [Online] 14. 01 2019. [Citace: 29. 11 2022.] https://blog.nrwl.io/understanding-angular-ivy-incremental-dom-and-virtual-dom-243be844bf36.
30. JavaScript. *GeeksforGeeks.* [Online] 29. 03 2022. [Citace: 22. 05 2022.] https://www.geeksforgeeks.org/javascript/?ref=ghm.
31. Haverbeke, Marijn. *Eloquent JavaScript: a modern introduction to programming.* 3.rd. . : No Starch Press, 2018. ISBN 9781593279516.
32. ECMA-262. *ECMA-INTERNATIONAL.* [Online] 1997-2021. [Citace: 24. 05 2022.] https://www.ecma-intenational.org/publications-and-standards/standards/ecma-262/.
33. Vanilla JS. *Vanilla JS.* [Online] [Citace: 14. 06 2022.] http://vanilla-js.com.
34. Drozdík, Martin. TYPESCRIPT, ANEB JAVASCRIPT NA DOBRÝCH DROGÁCH. *Bonsai Development.* [Online] 15. 05 2019. [Citace: 14. 06 2022.] https://bonsai-development.cz/clanek/typescript-aneb-javascript-na-dobrych-drogach.
35. Microsoft Docs. Přehled TypeScriptu. *Přehled TypeScriptu - Learn | Microsoft Docs.* [Online] [Citace: 14. 06 2022.] https://docs.microsoft.com/cs-cz/learn/modules/typescript-get-started/2-typescript-overview.
36. AMBLER, Tim a Nicholas CLOUD. *JavaScript frameworks for modern web dev.* New York : Apress, [2015]. Expert's voice in Web development. ISBN 14-842-0663-0..
37. GREIF, Sascha a Raphaël BENITTE. The State of JavaScript 2019. [Online] 23. 01 2019. [Citace: 14. 03 2022.] https://2019.stateofjs.com/.
38. What Are The Advantages and Disadvantages of React JS. *FASTCOMET.* [Online] 18. 03 2022. [Citace: 11. 08 2022.] https://www.fastcomet.com/blog/advantages-and-disadvantages-of-react-js#h-react-js-disadvantages.
39. You, Evan. Vue.js. [Online] © 2014-2022. [Citace: 13. 08 2022.] https://vuejs.org.
40. Vue.js Tutorial. *javaTpoint.* [Online] ©2011-2021. [Citace: 13. 08 2022.] https://www.javatpoint.com/vue-js.
41. *Angular.* [Online] Google, ©2010-2022. [Citace: 14. 08 2022.] https://angular.io.
42. Koffer, Peter. Comparison between Angular and React. *mDevelopers.* [Online] 18. 01 2022. [Citace: 2022. 08 14.] https://mdevelopers.com/blog/comparison-between-angular-and-react.

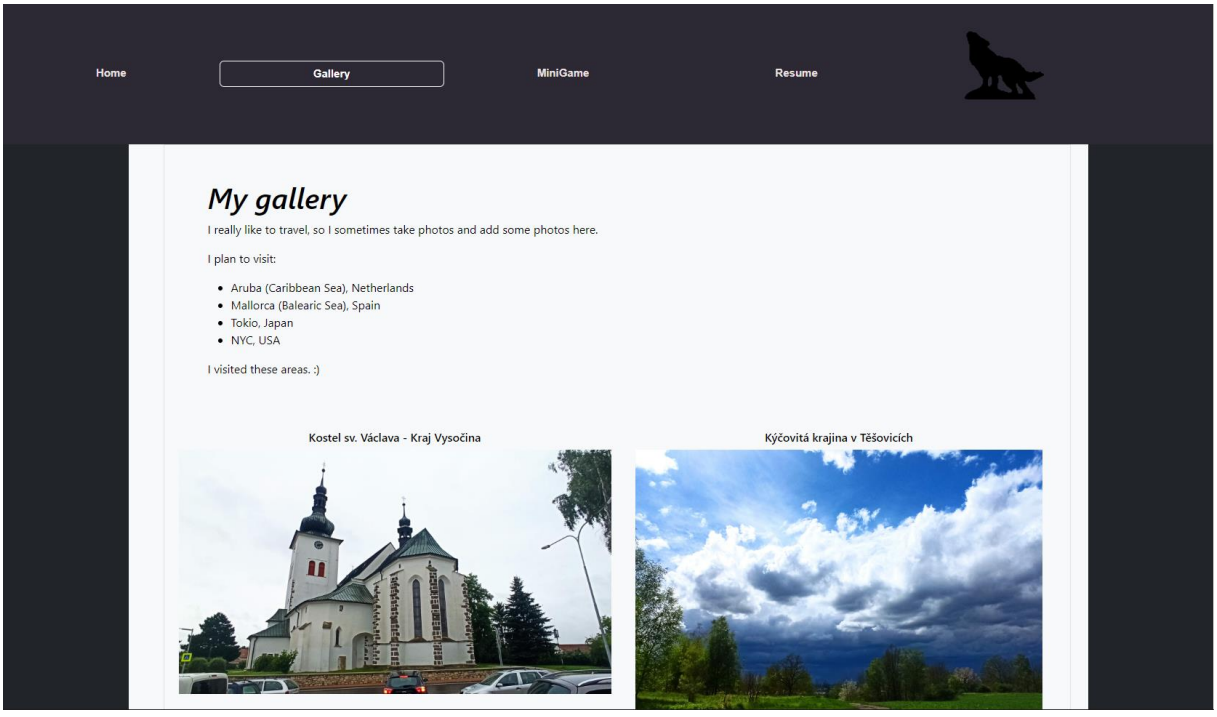
43. *The State of JS 2020.* [Online] 2020. [Citace: 29. 11 2022.] https://2020.stateofjs.com/en-US/.
44. *The state of JS 2021*. [Online] 2021. [Citace: 29. 11 2022.] https://2021.stateofjs.com/en-US/.
45. Stefan Krauseh. Results for js web frameworks benchmark - official run. *Official results for js web frameworks benchmark.* [Online] [Citace: 29. 11 2022.] https://krausest.github.io/js-framework-benchmark/2022/table_chrome_107.0.5304.62.html.
46. Vytváření javascriptových aplikací v Node.js. *Microsoft* . [Online] 2022. [Citace: 06. 11 2022.] https://learn.microsoft.com/cs-cz/training/paths/build-javascript-applications-nodejs/.
47. Máca, Jindřich. Úvod do Node.js. *itnetwork.* [Online] 18. 04 2018. [Citace: 06. 10 2022.] https://www.itnetwork.cz/javascript/nodejs/uvod-do-nodejs.
48. Node.js v18.10.0 documentation. *Node.js.* [Online] [Citace: 06. 10 2022.] https://nodejs.org/dist/latest-v18.x/docs/api/.
49. Janovský, Dušan. Jak stránky dostat na server. *Jak psat web.* [Online] ©1999 - 2022. [Citace: 18. 08 2022.] https://www.jakpsatweb.cz/publikovani.html.
50. POTÁČEK, Jiří. *Grafické uživatelské rozhraní.* [online] Praha : Národní knihpvna ČR, KTD: Česká terminologická databáze knihovnictví a informační vědy (TDKIV), 2003. Dostupné z: https://aleph.nkp.cz/F/?func=direct&doc_number=000000022&local_base=KTD.
51. Walton, Philip. First Contentful Paint (FCP). *Web.dev.* [Online] 07. 11 2019. [Citace: 03. 03 2023.] https://web.dev/fcp/.
52. Time to Interactive (TTI). *Web.dev.* [Online] 11. 05 2022. [Citace: 03. 03 2023.] https://web.dev/tti/.
53. Total Blocking Time (TBT). *Web.dev.* [Online] 11. 05 2022. [Citace: 03. 03 2023.] https://web.dev/tbt/.
54. Largest Contentful Paint (LCP). *Web.dev.* [Online] 19. 10 2022. [Citace: 03. 03 2023.] https://web.dev/lcp/.
55. Philip Walton, Milica Mihajlija. Cumulative Layout Shift (CLS). *Web.dev.* [Online] 19. 09 2022. [Citace: 03. 03 2023.] https://web.dev/cls/.

8 Přílohy

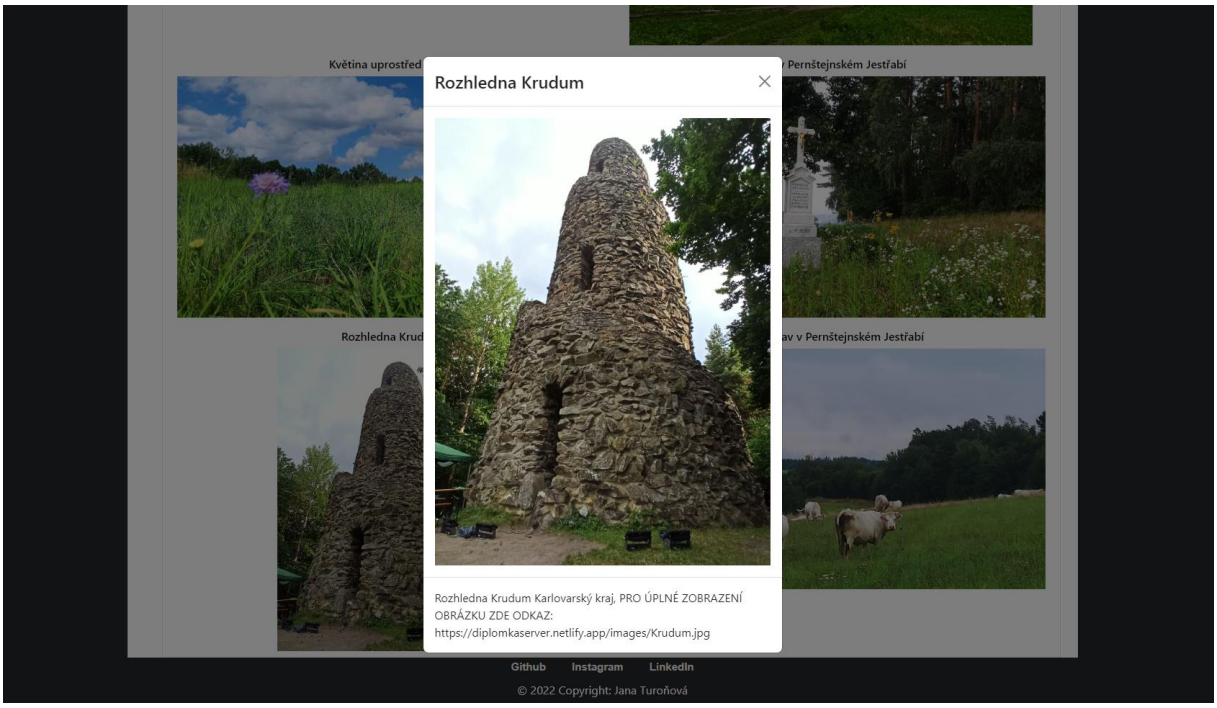
8.1 Ukázka webové stránky



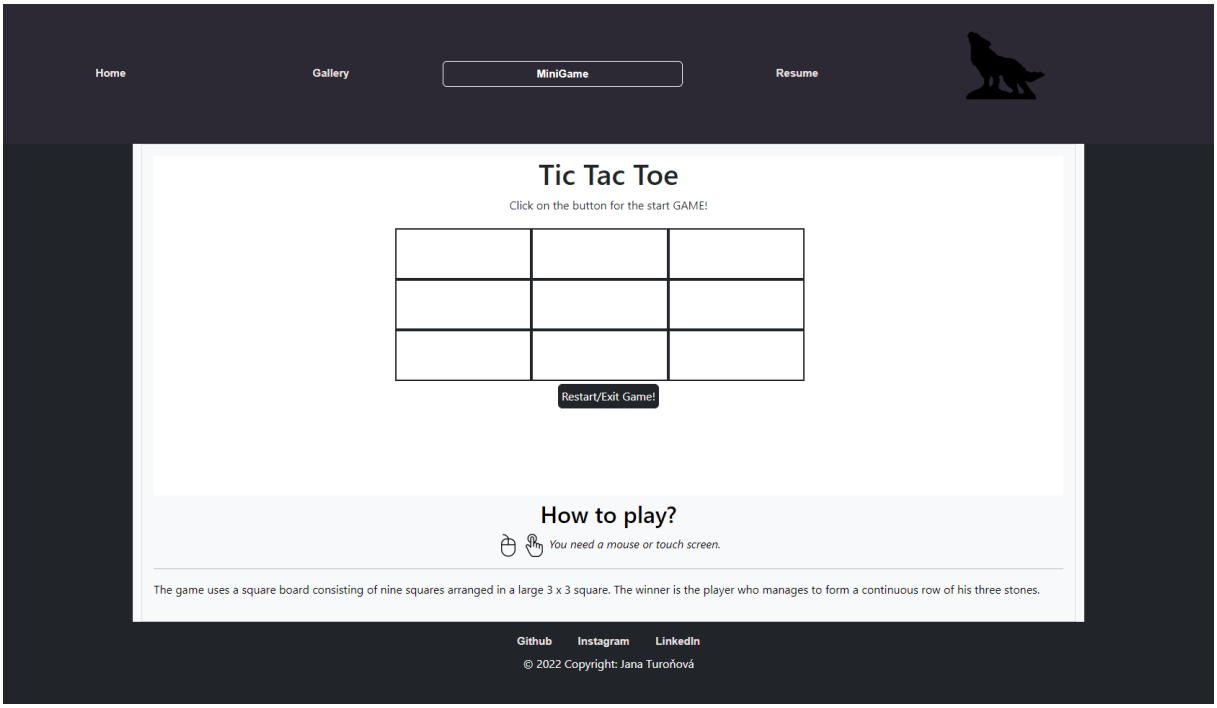
Obrázek 21- Domovská stránka (vlastní zpracování)



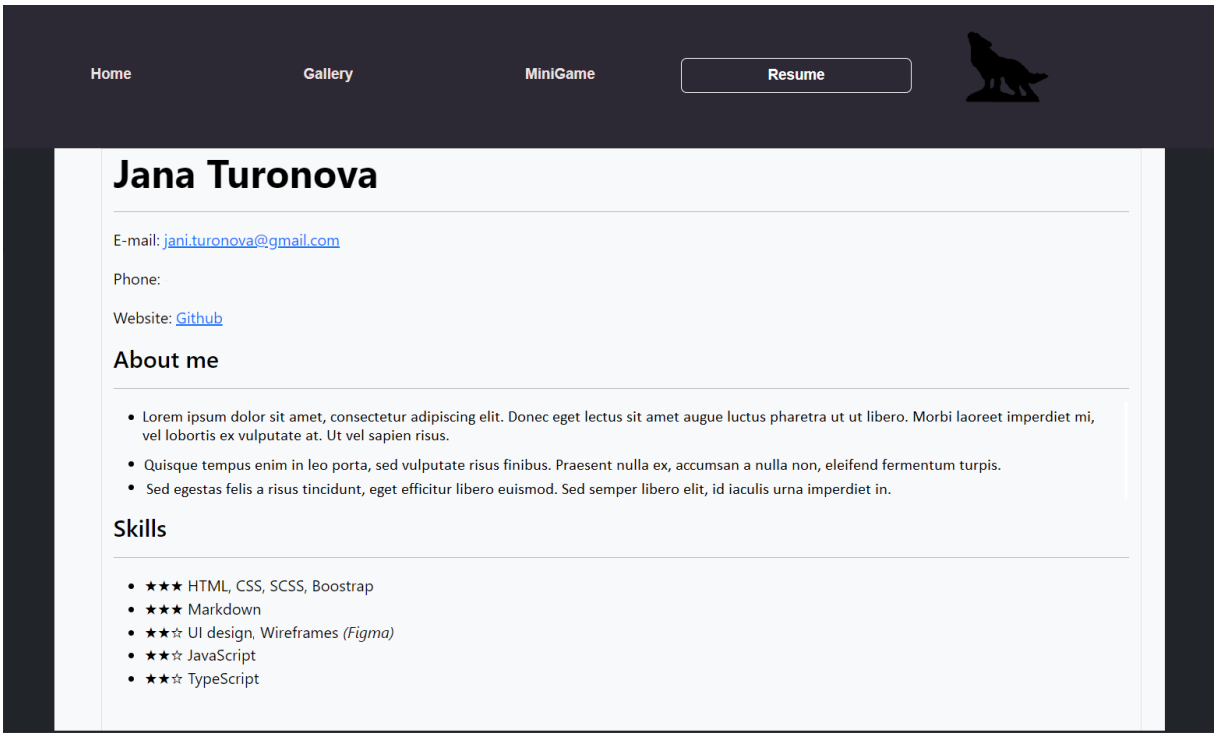
Obrázek 22 – Galerie (vlastní zpracování)



Obrázek 23 - Galerie – zvětšení obrázku (vlastní zpracování)



Obrázek 24 - Hra TicTacToe (vlastní zpracování)



Obrázek 25 – Resume (vlastní zpracování)

8.2 Měření

8.2.1 React – mobilní verze

Home		Gallery		MiniGame		Resume	
1. měření		1. měření		1. měření		1. měření	
First Contentful Paint	1.9	First Contentful Paint	1.9	First Contentful Paint	1.9	First Contentful Paint	1.9
Speed Index	2.0	Speed Index	2.5	Speed Index	2.1	Speed Index	2
Largest Contentful Paint	3.7	Largest Contentful Paint	3.1	Largest Contentful Paint	2.9	Largest Contentful Paint	2.9
Time to Interactive	2.3	Time to Interactive	2.3	Time to Interactive	2.4	Time to Interactive	2.4
Total Blocking Time	70	Total Blocking Time	80	Total Blocking Time	160	Total Blocking Time	130
Cumulative Layout Shift	0.45	Cumulative Layout Shift	0.081	Cumulative Layout Shift	0.042	Cumulative Layout Shift	0.108
76%		**91%**		**92%**		**91%**	
2. měření		2. měření		2. měření		2. měření	
First Contentful Paint	1.9	First Contentful Paint	1.9	First Contentful Paint	1.9	First Contentful Paint	1.9
Speed Index	2.8	Speed Index	2.2	Speed Index	1.9	Speed Index	1.9
Largest Contentful Paint	3.7	Largest Contentful Paint	6.1	Largest Contentful Paint	3.2	Largest Contentful Paint	2.9
Time to Interactive	2.1	Time to Interactive	2.4	Time to Interactive	2.3	Time to Interactive	2.3
Total Blocking Time	50	Total Blocking Time	170	Total Blocking Time	70	Total Blocking Time	60
Cumulative Layout Shift	0.31	Cumulative Layout Shift	0.265	Cumulative Layout Shift	0.038	Cumulative Layout Shift	0.108
79%		**67%**		**91%**		**92%**	
3. měření		3. měření		3. měření		3. měření	
First Contentful Paint	1.9	First Contentful Paint	1.9	First Contentful Paint	1.9	First Contentful Paint	1.9
Speed Index	2.3	Speed Index	1.9	Speed Index	2.7	Speed Index	2.2
Largest Contentful Paint	3.7	Largest Contentful Paint	3.2	Largest Contentful Paint	2.9	Largest Contentful Paint	2.9
Time to Interactive	2.3	Time to Interactive	2.3	Time to Interactive	3	Time to Interactive	2.4
Total Blocking Time	100	Total Blocking Time	100	Total Blocking Time	190	Total Blocking Time	100
Cumulative Layout Shift	0.31	Cumulative Layout Shift	0.081	Cumulative Layout Shift	0.038	Cumulative Layout Shift	0.108
78%		**90%**		**90%**		**91%**	
4. měření		4. měření		4. měření		4. měření	
First Contentful Paint	1.9	First Contentful Paint	1.9	First Contentful Paint	1.9	First Contentful Paint	1.9
Speed Index	2.3	Speed Index	2.7	Speed Index	2.7	Speed Index	2.3
Largest Contentful Paint	4.4	Largest Contentful Paint	3.2	Largest Contentful Paint	2.9	Largest Contentful Paint	3.1
Time to Interactive	2.3	Time to Interactive	2.4	Time to Interactive	2.4	Time to Interactive	2.4
Total Blocking Time	160	Total Blocking Time	170	Total Blocking Time	170	Total Blocking Time	180
Cumulative Layout Shift	0.31	Cumulative Layout Shift	0.110	Cumulative Layout Shift	0.038	Cumulative Layout Shift	0.108
72%		**88%**		**91%**		**88%**	
5. měření		5. měření		5. měření		5. měření	
First Contentful Paint	1.9	First Contentful Paint	1.9	First Contentful Paint	1.9	First Contentful Paint	1.9
Speed Index	2.3	Speed Index	2.1	Speed Index	1.9	Speed Index	1.9
Largest Contentful Paint	4.4	Largest Contentful Paint	3.1	Largest Contentful Paint	2.9	Largest Contentful Paint	2.9
Time to Interactive	2.3	Time to Interactive	2.4	Time to Interactive	2.4	Time to Interactive	2.5
Total Blocking Time	70	Total Blocking Time	140	Total Blocking Time	150	Total Blocking Time	270
Cumulative Layout Shift	0.51	Cumulative Layout Shift	0.249	Cumulative Layout Shift	0.038	Cumulative Layout Shift	0.108
70%		**83%**		**92%**		**86%**	

8.2.2 Vue – mobilní verze

Home		Gallery		MiniGame		Resume	
1. měření		1. měření		1. měření		1. měření	
First Contentful Paint	1.7	First Contentful Paint	1.7	First Contentful Paint	1.7	First Contentful Paint	1.7
Speed Index	2.2	Speed Index	2.5	Speed Index	2.5	Speed Index	2.6
Largest Contentful Paint	4.0	Largest Contentful Paint	2.7	Largest Contentful Paint	2.5	Largest Contentful Paint	2.5
Time to Interactive	1.9	Time to Interactive	2.1	Time to Interactive	2.3	Time to Interactive	1.8
Total Blocking Time	100	Total Blocking Time	80	Total Blocking Time	100	Total Blocking Time	70
Cumulative Layout Shift	0.489	Cumulative Layout Shift	0.161	Cumulative Layout Shift	0.141	Cumulative Layout Shift	0.19
	74%		**90%**		**93%**		**91%**
2. měření		2. měření		2. měření		2. měření	
First Contentful Paint	1.7	First Contentful Paint	1.7	First Contentful Paint	1.7	First Contentful Paint	1.7
Speed Index	2.5	Speed Index	2.6	Speed Index	1.9	Speed Index	2.4
Largest Contentful Paint	4.0	Largest Contentful Paint	2.6	Largest Contentful Paint	2.8	Largest Contentful Paint	2.5
Time to Interactive	2.1	Time to Interactive	2.0	Time to Interactive	2.3	Time to Interactive	1.9
Total Blocking Time	80	Total Blocking Time	70	Total Blocking Time	100	Total Blocking Time	110
Cumulative Layout Shift	0.489	Cumulative Layout Shift	0.161	Cumulative Layout Shift	0.141	Cumulative Layout Shift	0.19
	74%		**90%**		**91%**		**91%**
3. měření		3. měření		3. měření		3. měření	
First Contentful Paint	1.7	First Contentful Paint	1.7	First Contentful Paint	1.7	First Contentful Paint	1.7
Speed Index	1.8	Speed Index	3.1	Speed Index	2.6	Speed Index	2.5
Largest Contentful Paint	3.2	Largest Contentful Paint	2.9	Largest Contentful Paint	2.8	Largest Contentful Paint	2.5
Time to Interactive	1.9	Time to Interactive	2.0	Time to Interactive	2.2	Time to Interactive	1.9
Total Blocking Time	120	Total Blocking Time	90	Total Blocking Time	70	Total Blocking Time	100
Cumulative Layout Shift	0.218	Cumulative Layout Shift	0.211	Cumulative Layout Shift	0.141	Cumulative Layout Shift	0.19
	85%		**87%**		**91%**		**91%**
4. měření		4. měření		4. měření		4. měření	
First Contentful Paint	1.7	First Contentful Paint	1.7	First Contentful Paint	1.7	First Contentful Paint	1.7
Speed Index	2.1	Speed Index	1.7	Speed Index	3.0	Speed Index	1.7
Largest Contentful Paint	4	Largest Contentful Paint	2.7	Largest Contentful Paint	2.5	Largest Contentful Paint	2.5
Time to Interactive	1.9	Time to Interactive	2.0	Time to Interactive	2.1	Time to Interactive	1.8
Total Blocking Time	110	Total Blocking Time	80	Total Blocking Time	51	Total Blocking Time	80
Cumulative Layout Shift	0.218	Cumulative Layout Shift	0.402	Cumulative Layout Shift	0.141	Cumulative Layout Shift	0.19
	79%		**84%**		**93%**		**91%**
5. měření		5. měření		5. měření		5. měření	
First Contentful Paint	1.7	First Contentful Paint	1.7	First Contentful Paint	1.7	First Contentful Paint	1.7
Speed Index	1.7	Speed Index	1.7	Speed Index	1.7	Speed Index	2.5
Largest Contentful Paint	3.4	Largest Contentful Paint	4.3	Largest Contentful Paint	2.5	Largest Contentful Paint	2.5
Time to Interactive	1.9	Time to Interactive	2.0	Time to Interactive	2.2	Time to Interactive	2.1
Total Blocking Time	110	Total Blocking Time	77	Total Blocking Time	53	Total Blocking Time	160
Cumulative Layout Shift	0.218	Cumulative Layout Shift	0.460	Cumulative Layout Shift	0.141	Cumulative Layout Shift	0.19
84%			**72%**		**93%**		**89%**

8.2.3 Angular – mobilní verze

Home		Gallery		MiniGame		Resume	
1. měření		1. měření		1. měření		1. měření	
First Contentful Paint	2.1	First Contentful Paint	2.0	First Contentful Paint	2.1	First Contentful Paint	2
Speed Index	2.6	Speed Index	2.4	Speed Index	2.7	Speed Index	2.2
Largest Contentful Paint	4.4	Largest Contentful Paint	3.9	Largest Contentful Paint	3.3	Largest Contentful Paint	3.9
Time to Interactive	2.9	Time to Interactive	2.9	Time to Interactive	2.9	Time to Interactive	3.2
Total Blocking Time	430	Total Blocking Time	350	Total Blocking Time	360	Total Blocking Time	420
Cumulative Layout Shift	0.54	Cumulative Layout Shift	0.277	Cumulative Layout Shift	0.041	Cumulative Layout Shift	0.425
58%		**69%**		**82%**		**64%**	
2. měření		2. měření		2. měření		2. měření	
First Contentful Paint	2.1	First Contentful Paint	2.0	First Contentful Paint	2.1	First Contentful Paint	2
Speed Index	2.3	Speed Index	2.3	Speed Index	2.1	Speed Index	2.6
Largest Contentful Paint	4.6	Largest Contentful Paint	3.7	Largest Contentful Paint	3.7	Largest Contentful Paint	3.7
Time to Interactive	3.2	Time to Interactive	3.1	Time to Interactive	3.2	Time to Interactive	3.1
Total Blocking Time	470	Total Blocking Time	310	Total Blocking Time	310	Total Blocking Time	540
Cumulative Layout Shift	0.77	Cumulative Layout Shift	0.030	Cumulative Layout Shift	0.038	Cumulative Layout Shift	0.426
55%		**80%**		**80%**		**62%**	
3. měření		3. měření		3. měření		3. měření	
First Contentful Paint	2.1	First Contentful Paint	2.0	First Contentful Paint	2.1	First Contentful Paint	2
Speed Index	3.0	Speed Index	2.4	Speed Index	2.3	Speed Index	2.5
Largest Contentful Paint	4.0	Largest Contentful Paint	3.7	Largest Contentful Paint	3.9	Largest Contentful Paint	4.1
Time to Interactive	3.0	Time to Interactive	3.2	Time to Interactive	3.5	Time to Interactive	3.3
Total Blocking Time	500	Total Blocking Time	230	Total Blocking Time	400	Total Blocking Time	580
Cumulative Layout Shift	0.48	Cumulative Layout Shift	0.141	Cumulative Layout Shift	0.038	Cumulative Layout Shift	0.138
60%		**80%**		**76%**		**67%**	
4. měření		4. měření		4. měření		4. měření	
First Contentful Paint	2.0	First Contentful Paint	2.1	First Contentful Paint	2.1	First Contentful Paint	2
Speed Index	2.7	Speed Index	2.5	Speed Index	2.5	Speed Index	2.3
Largest Contentful Paint	4.6	Largest Contentful Paint	3.3	Largest Contentful Paint	3.7	Largest Contentful Paint	3.7
Time to Interactive	3.1	Time to Interactive	2.9	Time to Interactive	3.3	Time to Interactive	2.9
Total Blocking Time	500	Total Blocking Time	440	Total Blocking Time	450	Total Blocking Time	350
Cumulative Layout Shift	0.54	Cumulative Layout Shift	0.030	Cumulative Layout Shift	0.326	Cumulative Layout Shift	0.138
56%		**79%**		**66%**		**76%**	
5. měření		5. měření		5. měření		5. měření	
First Contentful Paint	2.1	First Contentful Paint	1.9	First Contentful Paint	2.1	First Contentful Paint	2.1
Speed Index	2.7	Speed Index	2.7	Speed Index	2.4	Speed Index	2.9
Largest Contentful Paint	3.8	Largest Contentful Paint	3.2	Largest Contentful Paint	3.5	Largest Contentful Paint	3.4
Time to Interactive	2.7	Time to Interactive	2.4	Time to Interactive	3.1	Time to Interactive	2.9
Total Blocking Time	340	Total Blocking Time	150	Total Blocking Time	260	Total Blocking Time	480
Cumulative Layout Shift	0.48	Cumulative Layout Shift	0.249	Cumulative Layout Shift	0.326	Cumulative Layout Shift	0.138
66%		**83%**		**74%**		**74%**	

8.2.4 React – desktopová verze

Home		Gallery		MiniGame		Resume	
1. měření		1. měření		1. měření		1. měření	
First Contentful Paint	0.5	First Contentful Paint	0.5	First Contentful Paint	0.5	First Contentful Paint	0.5
Speed Index	1	Speed Index	0.9	Speed Index	0.8	Speed Index	0.8
Largest Contentful Paint	1.0	Largest Contentful Paint	1.2	Largest Contentful Paint	0.7	Largest Contentful Paint	0.7
Time to Interactive	0.5	Time to Interactive	0.5	Time to Interactive	0.6	Time to Interactive	0.5
Total Blocking Time	0	Total Blocking Time	0	Total Blocking Time	0	Total Blocking Time	0
Cumulative Layout Shift	0.008	Cumulative Layout Shift	0.201	Cumulative Layout Shift	0.007	Cumulative Layout Shift	0.043
98%		**91%**		**100%**		**99%**	
2. měření		2. měření		2. měření		2. měření	
First Contentful Paint	0.5	First Contentful Paint	0.5	First Contentful Paint	0.6	First Contentful Paint	0.5
Speed Index	0.8	Speed Index	0.8	Speed Index	0.8	Speed Index	0.9
Largest Contentful Paint	1.0	Largest Contentful Paint	1.4	Largest Contentful Paint	0.7	Largest Contentful Paint	0.7
Time to Interactive	0.5	Time to Interactive	0.5	Time to Interactive	0.6	Time to Interactive	0.5
Total Blocking Time	0	Total Blocking Time	0	Total Blocking Time	0	Total Blocking Time	0
Cumulative Layout Shift	0.004	Cumulative Layout Shift	0.200	Cumulative Layout Shift	0.007	Cumulative Layout Shift	0.043
99%		**90%**		**100%**		**99%**	
3. měření		3. měření		3. měření		3. měření	
First Contentful Paint	0.5	First Contentful Paint	0.6	First Contentful Paint	0.6	First Contentful Paint	0.5
Speed Index	0.6	Speed Index	0.9	Speed Index	0.7	Speed Index	1.1
Largest Contentful Paint	1	Largest Contentful Paint	1.7	Largest Contentful Paint	0.7	Largest Contentful Paint	0.7
Time to Interactive	0.5	Time to Interactive	0.5	Time to Interactive	0.6	Time to Interactive	0.5
Total Blocking Time	0	Total Blocking Time	0	Total Blocking Time	0	Total Blocking Time	0
Cumulative Layout Shift	0.004	Cumulative Layout Shift	0.144	Cumulative Layout Shift	0.007	Cumulative Layout Shift	0.043
99%		**90%**		**100%**		**99%**	
4. měření		4. měření		4. měření		4. měření	
First Contentful Paint	0.5	First Contentful Paint	0.5	First Contentful Paint	0.5	First Contentful Paint	0.5
Speed Index	0.7	Speed Index	0.7	Speed Index	0.8	Speed Index	0.8
Largest Contentful Paint	1	Largest Contentful Paint	2.4	Largest Contentful Paint	0.7	Largest Contentful Paint	0.7
Time to Interactive	1	Time to Interactive	0.5	Time to Interactive	0.7	Time to Interactive	0.5
Total Blocking Time	0	Total Blocking Time	0	Total Blocking Time	0	Total Blocking Time	0
Cumulative Layout Shift	0.005	Cumulative Layout Shift	0.229	Cumulative Layout Shift	0.007	Cumulative Layout Shift	0.043
98%		**81%**		**100%**		**99%**	
5. měření		5. měření		5. měření		5. měření	
First Contentful Paint	0.6	First Contentful Paint	0.5	First Contentful Paint	0.5	First Contentful Paint	0.5
Speed Index	0.6	Speed Index	0.8	Speed Index	0.5	Speed Index	0.5
Largest Contentful Paint	1	Largest Contentful Paint	2.1	Largest Contentful Paint	0.7	Largest Contentful Paint	0.7
Time to Interactive	0.6	Time to Interactive	0.5	Time to Interactive	0.5	Time to Interactive	0.5
Total Blocking Time	0	Total Blocking Time	0	Total Blocking Time	0	Total Blocking Time	0
Cumulative Layout Shift	0.008	Cumulative Layout Shift	0.181	Cumulative Layout Shift	0.008	Cumulative Layout Shift	0.043
99%		**85%**		**100%**		**100%**	

8.2.5 Vue – desktopová verze

Home		Gallery		MiniGame		Resume	
1. měření		1. měření		1. měření		1. měření	
First Contentful Paint	0.4	First Contentful Paint	0.4	First Contentful Paint	0.4	First Contentful Paint	0.4
Speed Index	0.7	Speed Index	0.9	Speed Index	1	Speed Index	0.8
Largest Contentful Paint	0.9	Largest Contentful Paint	1	Largest Contentful Paint	0.7	Largest Contentful Paint	0.7
Time to Interactive	0.4	Time to Interactive	0.4	Time to Interactive	0.4	Time to Interactive	0.5
Total Blocking Time	0	Total Blocking Time	0	Total Blocking Time	0	Total Blocking Time	40
Cumulative Layout Shift	0.003	Cumulative Layout Shift	0.106	Cumulative Layout Shift	0.06	Cumulative Layout Shift	0.087
99%		**97%**		**99%**		**98%**	
2. měření		2. měření		2. měření		2. měření	
First Contentful Paint	0.4	First Contentful Paint	0.4	First Contentful Paint	0.4	First Contentful Paint	0.4
Speed Index	0.5	Speed Index	0.7	Speed Index	0.8	Speed Index	0.6
Largest Contentful Paint	0.9	Largest Contentful Paint	1.1	Largest Contentful Paint	0.7	Largest Contentful Paint	0.6
Time to Interactive	0.4	Time to Interactive	0.4	Time to Interactive	0.4	Time to Interactive	0.4
Total Blocking Time	0	Total Blocking Time	0	Total Blocking Time	0	Total Blocking Time	0
Cumulative Layout Shift	0.003	Cumulative Layout Shift	0.137	Cumulative Layout Shift	0.06	Cumulative Layout Shift	0.087
99%		**95%**		**99%**		**99%**	
3. měření		3. měření		3. měření		3. měření	
First Contentful Paint	0.4	First Contentful Paint	0.4	First Contentful Paint	0.4	First Contentful Paint	0.4
Speed Index	0.8	Speed Index	0.9	Speed Index	0.8	Speed Index	0.9
Largest Contentful Paint	0.9	Largest Contentful Paint	1.6	Largest Contentful Paint	0.7	Largest Contentful Paint	0.7
Time to Interactive	0.5	Time to Interactive	0.5	Time to Interactive	0.4	Time to Interactive	0.5
Total Blocking Time	10	Total Blocking Time	0	Total Blocking Time	0	Total Blocking Time	20
Cumulative Layout Shift	0.003	Cumulative Layout Shift	0.152	Cumulative Layout Shift	0.06	Cumulative Layout Shift	0.087
99%		**91%**		**99%**		**98%**	
4. měření		4. měření		4. měření		4. měření	
First Contentful Paint	0.4	First Contentful Paint	0.4	First Contentful Paint	0.4	First Contentful Paint	0.4
Speed Index	0.4	Speed Index	1.0	Speed Index	0.9	Speed Index	0.6
Largest Contentful Paint	1	Largest Contentful Paint	2.2	Largest Contentful Paint	0.7	Largest Contentful Paint	0.7
Time to Interactive	0.5	Time to Interactive	0.4	Time to Interactive	0.4	Time to Interactive	0.4
Total Blocking Time	10	Total Blocking Time	0	Total Blocking Time	0	Total Blocking Time	0
Cumulative Layout Shift	0.003	Cumulative Layout Shift	0.160	Cumulative Layout Shift	0.06	Cumulative Layout Shift	0.087
99%		**85%**		**99%**		**99%**	
5. měření		5. měření		5. měření		5. měření	
First Contentful Paint	0.4	First Contentful Paint	0.4	First Contentful Paint	0.4	First Contentful Paint	0.4
Speed Index	0.8	Speed Index	1.0	Speed Index	0.7	Speed Index	1
Largest Contentful Paint	0.9	Largest Contentful Paint	1.8	Largest Contentful Paint	0.7	Largest Contentful Paint	0.7
Time to Interactive	0.5	Time to Interactive	0.5	Time to Interactive	0.5	Time to Interactive	0.5
Total Blocking Time	30	Total Blocking Time	10	Total Blocking Time	10	Total Blocking Time	20
Cumulative Layout Shift	0.003	Cumulative Layout Shift	0.130	Cumulative Layout Shift	0.061	Cumulative Layout Shift	0.087
99%		**90%**		**99%**		**98%**	

8.2.6 Angular – desktopová verze

Home		Gallery		MiniGame		Resume	
1. měření		1. měření		1. měření		1. měření	
First Contentful Paint	0.5	First Contentful Paint	0.5	First Contentful Paint	0.5	First Contentful Paint	0.5
Speed Index	1	Speed Index	1	Speed Index	0.7	Speed Index	0.9
Largest Contentful Paint	1.1	Largest Contentful Paint	2.1	Largest Contentful Paint	0.9	Largest Contentful Paint	0.9
Time to Interactive	0.7	Time to Interactive	0.6	Time to Interactive	0.8	Time to Interactive	0.7
Total Blocking Time	30	Total Blocking Time	40	Total Blocking Time	70	Total Blocking Time	30
Cumulative Layout Shift	0.004	Cumulative Layout Shift	0.157	Cumulative Layout Shift	0.006	Cumulative Layout Shift	0.063
98%		**86%**		**99%**		**99%**	
2. měření		2. měření		2. měření		2. měření	
First Contentful Paint	0.5	First Contentful Paint	0.5	First Contentful Paint	0.5	First Contentful Paint	2
Speed Index	0.7	Speed Index	0.7	Speed Index	1.9	Speed Index	2.2
Largest Contentful Paint	1.1	Largest Contentful Paint	1.3	Largest Contentful Paint	1	Largest Contentful Paint	3.6
Time to Interactive	0.7	Time to Interactive	0.7	Time to Interactive	0.8	Time to Interactive	2.7
Total Blocking Time	30	Total Blocking Time	30	Total Blocking Time	120	Total Blocking Time	210
Cumulative Layout Shift	0.127	Cumulative Layout Shift	0.101	Cumulative Layout Shift	0.006	Cumulative Layout Shift	0.138
95%		**95%**		**93%**		**82%**	
3. měření		3. měření		3. měření		3. měření	
First Contentful Paint	0.5	First Contentful Paint	0.5	First Contentful Paint	0.5	First Contentful Paint	0.5
Speed Index	0.7	Speed Index	0.6	Speed Index	0.7	Speed Index	0.7
Largest Contentful Paint	1.1	Largest Contentful Paint	1.3	Largest Contentful Paint	0.9	Largest Contentful Paint	1
Time to Interactive	0.6	Time to Interactive	0.7	Time to Interactive	0.7	Time to Interactive	0.7
Total Blocking Time	20	Total Blocking Time	30	Total Blocking Time	30	Total Blocking Time	50
Cumulative Layout Shift	0.004	Cumulative Layout Shift	0.356	Cumulative Layout Shift	0.006	Cumulative Layout Shift	0.063
98%		**87%**		**99%**		**98%**	
4. měření		4. měření		4. měření		4. měření	
First Contentful Paint	0.5	First Contentful Paint	0.5	First Contentful Paint	0.8	First Contentful Paint	0.5
Speed Index	0.8	Speed Index	0.7	Speed Index	0.9	Speed Index	0.7
Largest Contentful Paint	1.1	Largest Contentful Paint	2.9	Largest Contentful Paint	0.9	Largest Contentful Paint	0.9
Time to Interactive	0.6	Time to Interactive	0.7	Time to Interactive	0.8	Time to Interactive	0.8
Total Blocking Time	30	Total Blocking Time	30	Total Blocking Time	0	Total Blocking Time	30
Cumulative Layout Shift	0.004	Cumulative Layout Shift	0.086	Cumulative Layout Shift	0.006	Cumulative Layout Shift	0.063
98%		**83%**		**99%**		**99%**	
5. měření		5. měření		5. měření		5. měření	
First Contentful Paint	0.5	First Contentful Paint	0.5	First Contentful Paint	0.5	First Contentful Paint	0.5
Speed Index	0.5	Speed Index	0.8	Speed Index	0.5	Speed Index	0.7
Largest Contentful Paint	1.1	Largest Contentful Paint	1.9	Largest Contentful Paint	0.9	Largest Contentful Paint	1.3
Time to Interactive	0.7	Time to Interactive	0.6	Time to Interactive	0.7	Time to Interactive	1
Total Blocking Time	100	Total Blocking Time	40	Total Blocking Time	30	Total Blocking Time	60
Cumulative Layout Shift	0.127	Cumulative Layout Shift	0.129	Cumulative Layout Shift	0.008	Cumulative Layout Shift	0.098
94%		**89%**		**99%**		**95%**	